April 2012

Empowerment Simulation System

Timothy John Kolek
Worcester Polytechnic Institute

Project Number: DC MQP 1102

Empowerment Simulation System

April 23, 2012

A Major Qualifying Project Report:

Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science in

Interactive Media and Game Development

By:

Timothy Kolek

Advised by:

Professor David Cyganski

Abstract

Empowerment is a new game being developed by WPI and FIRST, inspired by the FIRST Robotics Competition, aimed at educating high school students about Energy. The project goal was creation of a sandbox for game physics and GUI exploration. It allows a player to configure electrical power networks using 3D models of various (and adjustable) types of generators, consumers, transmission towers and interconnecting power lines. The simulator demonstrates the automatic solution for power distribution, a core component of game state, based upon system parameters and player choices. The simulator was built upon the JMonkey 3 game engine in Java, yielding an operating system agnostic, open source implementation as a basis for the future game system implementation.

Table of Contents

Abstract 2

Table of Contents 3

Table of Figures 6

1 Introduction 8

1.1 WPI Involvement 8

1.2 Empowerment Group 9

1.3 Empowerment Competition 9

1.4 Competition Focus 10

2 The Empowerment Game 12

2.1 Pre-Competition 12

2.2 Competition Day – Phase 1 13

2.3 Competition Day – Phase 2 17

2.4 Empowerment vs. FIRST Robotics 18

3 The Role of This MQP Project 20

4 Game Display Design Requirements 21

4.1 Different Structures 21

4.2 More Information 23

5 Components of the Empowerment Prototype 24

5.1 Game Engine Possibilities 24

5.2 Deciding on an Engine 26

5.3 JMonkey Engine 27

6 Experiences 29

6.1 Problems 29

6.2 Beta Version 30

6.3 Object Selection 31

7 Prototype Design 32

7.1 Controls 32

7.2 3D Models and Textures 46

7.3 Example Application 53

8 Conclusion 57

9 Appendices 59

9.1 Appendix A: Framework Analysis 59

9.2 Appendix B: XML Example 61

Table of Figures

Figure 1: Graph of Sources and Uses of Energy (From U.S. Energy Information Administration / Annual Energy Review 2009, DOE/EIA-0384(2009)) 10

Figure 2: Image of the Empowerment Grid 14

Figure 3: A possible grid. 14

Figure 4: A second possible grid. 15

Figure 5: A third possible grid. 15

Figure 6: A grid with objects and connections. 16

Figure 7: Leaning tower 31

Figure 8: The Start-up (Splash) Screen 33

Figure 9: Instructions Screen 34

Figure 10: Game World Screen 34

Figure 11: Splash screen showing panels 36

Figure 12: On left clicking an object, selection is indicated by its 3D representation becoming a blue wireframe model. The object will now move with the mouse. 38

Figure 13: Once the selected object has been moved, it is de-selected by releasing the left mouse button upon which its solid representation reappears. 39

Figure 14: On right clicking an object, selection is indicated by its 3D representation becoming an orange wireframe model. A transmission line will be created between this and the next selected object. 40

Figure 15: Once a second object has been selected, the solid representation reappears and a wire is created connecting the two objects. 41

Figure 16: Before updating the object, the max power is seen as 10 MW. 42

Figure 17: After setting the slider to a value and updating the object, the object's max power is 43 MW. 43

Figure 18: The connection between the towers is incapable of supporting the level of power demanded by consumers. This is indicated by its color turning red. 44
Figure 19: A consumer is not receiving its required power. This is indicated by the brownout percentage. 45
Figure 20: Example of a solved network with no problems 45
Figure 21: Coal Plant Object Model 47
Figure 22: Oil Plant Object Model 47
Figure 23: Natural Gas Plant Object Model 48
Figure 24: Hydroelectric Dam Object Model 48
Figure 25: Solar Panel Object Model 49
Figure 26: Wind Turbine Object Model 50
Figure 27: Nuclear Power Plant Object Model 50
Figure 28: House Object Model 51
Figure 29: Building Object Model 51
Figure 30: Skyscraper Object Model 52
Figure 31: Transmission Tower Object Model 52
Figure 32: Objects added 53
Figure 33: Connections added 54
Figure 34: Network solved 54
Figure 35: Wire Capacity Reached 55
Figure 36: Multiple Brownouts in Network 56

1 Introduction

FIRST is a non-profit organization founded by Dean Kamen. It was created specifically to support the FIRST Robotics Competition (FRC). Kamen is also the president of the organization called DEKA. This is his for profit organization. FIRST stands for For Inspiration and Recognition of Science and Technology. The FIRST Robotics Competition is a pretty large event. In 2011, the number of participating students was more than 248,000; more than 66,000 mentors and adult supporters participated as part of the teams; there were 22,475 teams in the competition and these teams created 20,675 robots.

1.1 WPI Involvement

WPI became involved in developing Empowerment because of a challenge put forth by Google and Dean Kamen. This challenge was to create a game that could teach the general population about energy and the policies involving it. The game needed to inform people and empower them (hence the name Empowerment). Another quality was that the game should be fun and attractive for students so people would follow the example put forth from the ideas instilled in the game that will hopefully be taken away by the players. Ideally, this would create a new subculture of people that were informed about energy. A final quality Google and Kamen wanted was to create a connection with the officials who made the energy policies. It should develop an interest by providing opportunities for the media to showcase the game and the policies regarding energy along with it. This would make the ideas about energy become a reality to more people. In the end, this project could change how the world thinks.

1.2 Empowerment Group

Empowerment is being created by three people at WPI. The first is Professor David Cyganski. He is a professor in the Electrical and Computer Engineering Department. He was chosen for his skills in electrical engineering which would be needed for the real world aspects of the Empowerment competition. Since the challenge was to create a game, someone with experience in the video games industry, specifically a designer, was also needed. This second person is Professor Brian Moriarty. He is a professor in the Interactive Media and Game Development Department. The third person is Dan Tennant who is an Interactive Media and Game Development research assistant. He is aiding the professors in this project by completing any research needed in developing the Empowerment project.

1.3 Empowerment Competition

Empowerment is to be a large scale competition in a vein similar to the FIRST Robotics competition. Just as the FIRST Robotics Competition is aimed toward high school students interested in robotics and engineering, Empowerment will be aimed toward high school students interested in the main topics of the event. While the FIRST Robotics Competition focuses on creating robots, Empowerment focuses on energy, its usage, and its many different forms. The emphasis is on education and an inspiration for students to learn more about energy.

As the idea for Empowerment developed, multiple proposals were made. Each of these proposals had three parts: an arena play component, a social network component, and a PC game component. The proposals had differing focuses, with each component taking up a different portion of the entire competition. There have also been three large revisions of the idea since it was proposed. These revisions took place after a focus group review, a brainstorming session with multiple WPI departments, and a preliminary design review. The development and design

of the Empowerment competition continues to through the end of this project and will continue thereafter for months to come.

1.4 Competition Focus

Since energy is a very general topic, a focus had to be created for the actual competition. After analyzing different aspects of energy generation and consumption, the focus chosen was the electric grid. This seemed to be the most readily playable aspect of the topic of energy. The playable version of the game could incorporate the sources of energy, the uses of energy, how energy is moved around geographically, and how energy is regulated. More possibilities are the different resources needed as different types of sources of energy. Converting energy into different forms is yet another possibility of something else to implement into the game.

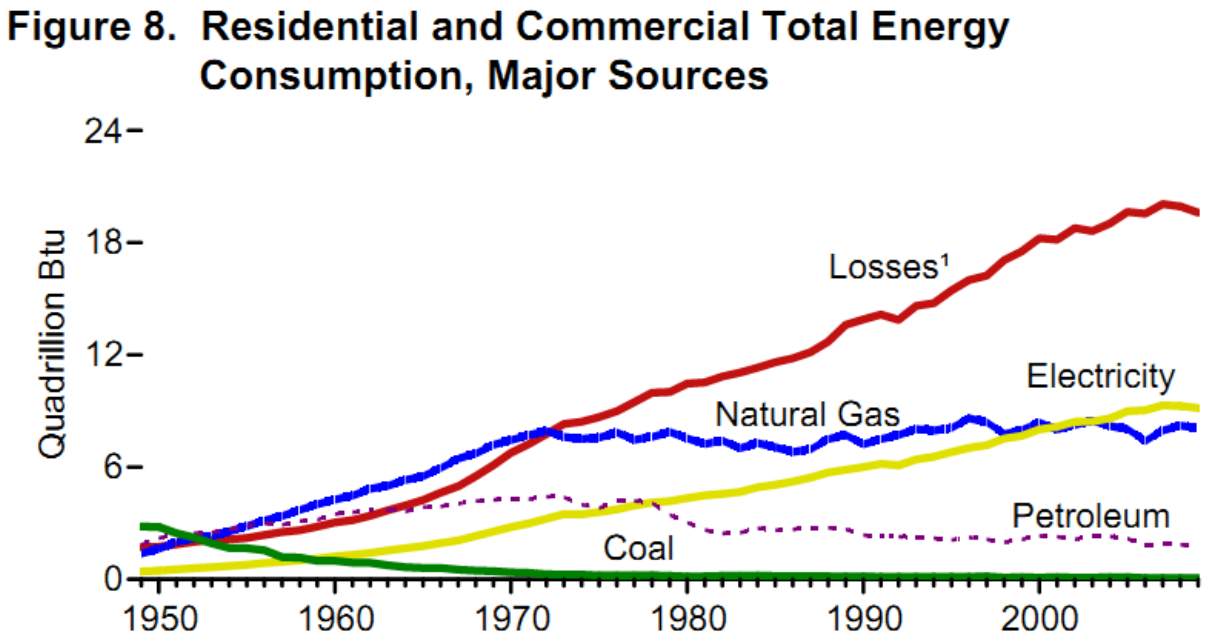

Figure 1: Graph of Sources and Uses of Energy (From U.S. Energy Information Administration / Annual Energy Review 2009, DOE/EIA-0384(2009))

As shown by the figure above, the current electrical power grid is very inefficient. The energy lost is approximately double of the power society gets from electricity or natural gas alone. The Empowerment game could educate people about these facts and help them to create a more efficient system so society can get the most out of the energy that is generated.

2 The Empowerment Game

There are multiple parts to the Empowerment competition. The opening activities will take place in school science classes. The curriculum will be modified to allow for the students to learn about different forms of energy and how they differ. Different means of generating energy such as hydroelectric power or wind power have different positives and negatives associated with them. Wind turbines rely on the wind so it varies greatly depending on the speed of the wind. The same can be said about hydroelectric power. The flow of water determines how well the dam will work generating power from the running water. The game is also about the different ways of generating power such as burning coal or using radioactive elements to generate energy. The environmental impact of different types of energy is also a part of Empowerment. Nuclear power thermally contaminates a lot of water that is used as a coolant in the plants. Wind turbines have to be built in generally windy areas which can encroach upon the habitat of different animal species. This is an issue for example for the current proposal to build some wind turbine farms off the coast of Cape Cod. Environmentalists are concerned about drastically changing the territories of animals even to make a cleaner form of energy. Even solar and wind can be viewed as not clean forms of energy because they are still damaging the environment even though it is a problem with the locations and not the energy generation or the use of it.

2.1 Pre-Competition

This beginning phase of the game in the school curriculum is in preparation for the actual competition. In the Research and Development phase, students form teams and choose from multiple possibilities for which activities they choose to focus on. They could physically build a miniature version of an actual generator. Another option is to have a debate about legal policies or other controversial, energy related topics. A third option is to compile a research project such

as an examination of the effects of building wind turbines on local plant and animal life or how the building of a hydroelectric dam can uproot animals from their habitats and force people into moving out of their homes. There are infinite possibilities only limited by the imagination of the students. Each year, the competition will have a specific energy theme which each of these Research and Development projects would need to follow and connect to in some fashion. At the actual competition, the teams will discuss what they learned and the results of their projects with the judges. Before the competition date, the teams will be able to download simulation software for free to help them to learn how to create electrical power networks. The students will use this to learn the efficiencies of different types of energy generation and how to most cheaply and reliably deliver energy to consumers. Playing this embedded game is basically practice for the main portion of the competition. The teams can learn the ups and downs of all the elements and develop different strategies for creating the network. As will be explained below, this MQP is most connected to this embedded video game's development.

2.2 Competition Day – Phase 1

At the competition, two separate phases take place. The first phase involves competitions between individual teams composed of students from the participating high schools. The teams will show their presentations to the VIPs or stage what happened during their debate(s) for them. Miniatures are tested in chambers for how effectively they work in a multitude of situations pertinent to the form of energy being used and tested as the theme for the year's competition. While other groups are showing off their work, teams have their scrimmages at the same time to condense the total amount of time the entire competition will take to complete.

These scrimmages take place on the official game board. This is made up of two large (30 foot by 24 foot) grids made of LED strips in a specific pattern. Certain edges on the grid can

be effectively “turned-off” as to not allow these to be part of play for a given session. This creates a large number of possible grids that can be created and used. Many different grids can be made by eliminating different connections in this grid.

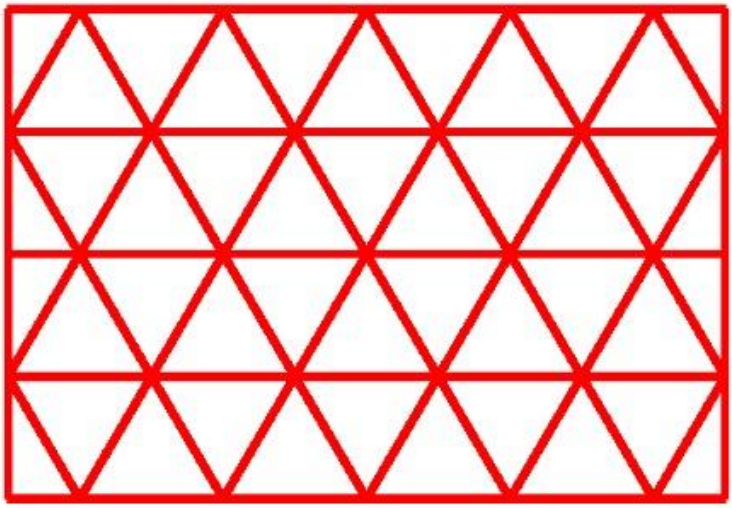

Figure 2: Image of the Empowerment Grid

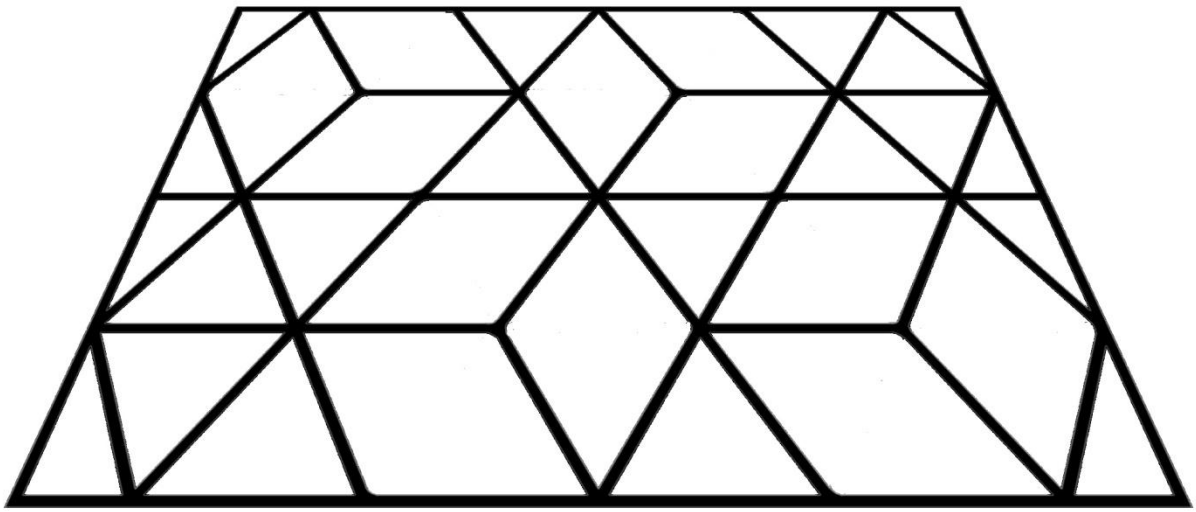

Figure 3: A possible grid.

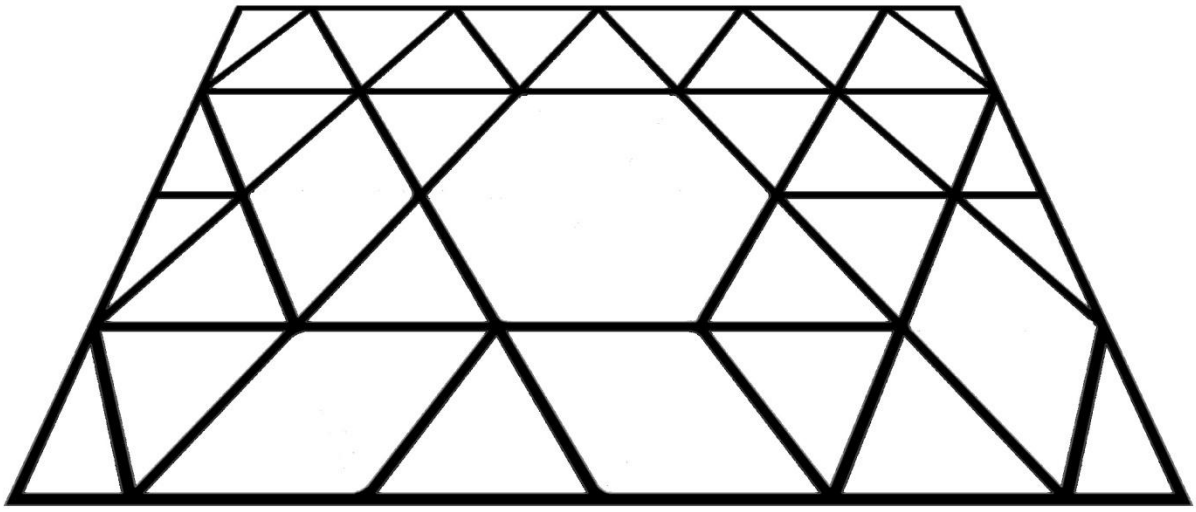

Figure 4: A second possible grid.

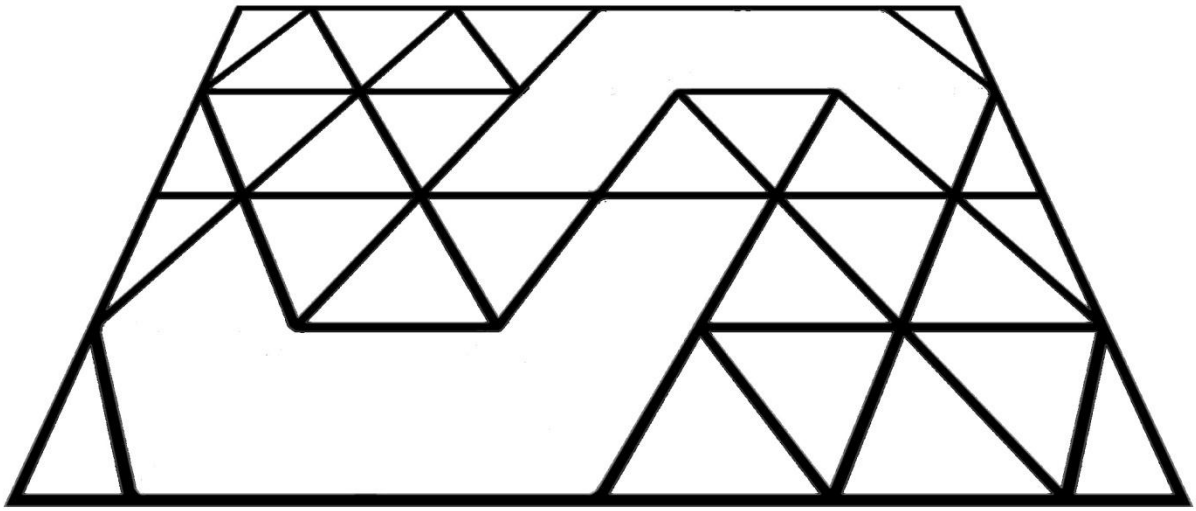

Figure 5: A third possible grid.

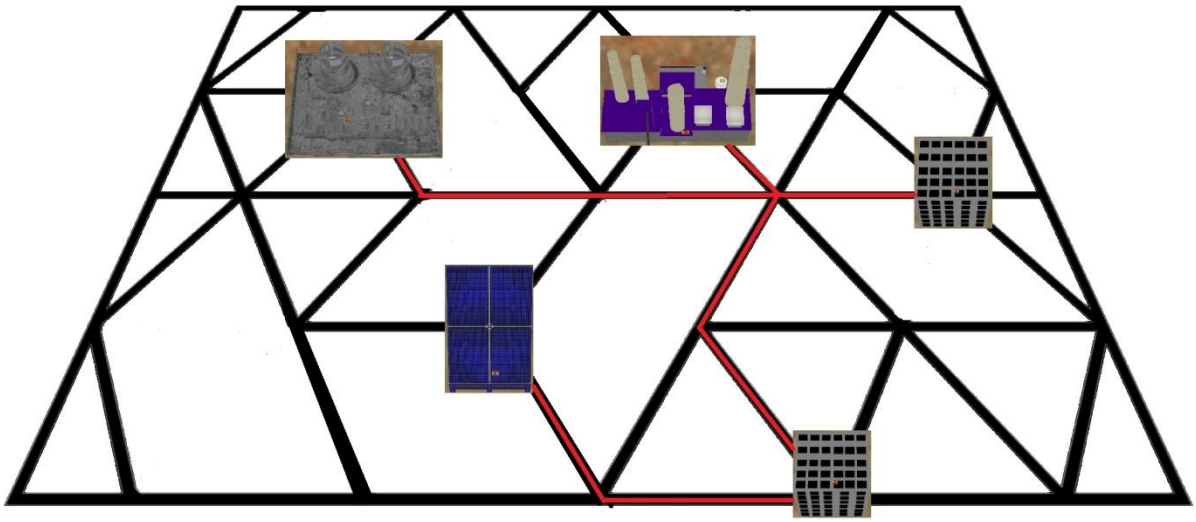

Figure 6: A grid with objects and connections.

On a projection screen above the main gameplay area seen by the audience and on the computer monitors used by the players, the grid will be overlaid on top of an actual image of terrain that corresponds to the open connections on the grid. For example, a mountain could exist over which no connections could be made or it is very expensive to do so. On the grid, each team builds an energy distribution network using the existing consumers placed by the organizers which are different sized cities. Each has a power need and the teams use generators to create the power and transmission lines (activated edges on the Empowerment grid) to transfer power to the cities by connecting them with transmission lines and towers. Powering the grids will also be accomplished through muscle power of teammates. Some members could ride on stationary bikes, others could power hand cranks, and still others could run on treadmills. Their output of physical energy will be converted to energy for use in the game. During the course of the scrimmage, natural disasters will occur. To overcome these obstacles, teams will have to complete mental and physical challenges that correspond to problems that can occur from the type of disaster that happened.

2.3 Competition Day – Phase 2

In the second phase of the competition, scores from the scrimmages and judges' opinions and other evaluations are used to resolve the positions (or rank) of teams in the tournament. Instead of competitions between individual teams, this phase has alliances forming the two sides of the competition. These alliances are created from multiple teams working together. The rounds will be between alliances playing the same type of game from the team scrimmages. Power in these rounds comes from the same physical generators. In addition, generators are awarded to teams who did well in the team scrimmages and different presentations from phase 1 and some other forms of competitive acquisitions. These various additional forms of acquisitions could be changing each year of the competition according to the energy focus.

Scoring in all of the competition rounds comes from a variety of sources. Physical energy from the people generating energy will be converted into points. The more efficient a power network is, the more points awarded to the team. Power outages remove points from a team's score according to the amount of outages and how long power was out. Points will also be given for challenges that may occur during the games.

Just as in FIRST Robotics, there is an audience present to watch the completion. The difference is that Empowerment audience members will have the ability to participate. Smart phones will be used to invest imaginary money in different teams as if they were companies. The audience can create an investment portfolio of multiple teams to try to maximize their profit. The best investor in the audience will be presented with an award at the end of the entire competition. Teams are also allowed to invest in themselves or other teams to make more imaginary money which can contribute to their final scores.

2.4 Empowerment vs. FIRST Robotics

The Empowerment competition is much more diverse that the FIRST Robotics Competition in the interests that could entice students to participate. The most obvious potential participants are students with an interest in energy. The entire competition involves energy. Students with an interest in engineering can also get involved with the physical building of a generator. Environment-interested students can look into the best energy technology choices for the environment or effects on it. Debates and research into laws and policies can attract students interested in politics. The business of selling and buying power can interest business-oriented students. Current events are also involved considering the current conflicts over energy policies. Students who like debate can become involved. Athletes can be the muscle for the power generators on the side. Students with an interest in video games can help creating strategies and aid in the playing of this "game". The potentially massive number of students that could become part of this competition dwarfs the smaller number of students that can become interested and involved in FIRST Robotics which is mainly those with interest in computer science and engineering.

Another disparity between the competitions is the cost to become a part of the game. FIRST Robotics participation means a team needs to purchase robotics kits and find a skilled engineer to be a mentor. If a teacher wants to become a part of this, there is a lot of training needed. Empowerment uses free game software for practice before the competition. Courses in the school can be modified to aid in the learning of energy. The building of miniature generators can be used as science fair projects. The final plus is that no highly skilled mentor is needed for anything in Empowerment. Overall, Empowerment is more accessible to more students and

schools because of the diverse interests that can apply to the competition and how cheap it is to get started.

3 The Role of This MQP Project

When the audience is watching the competition between teams and alliances, it can be difficult to see what is happening on the floor where the teams are actually playing and manipulating the system. Even being able to see the action occurring on the floor does not guarantee it will make sense to every audience member. A representation of the power network each team is creating should instead be represented on a screen that will create a more realistic portrayal of the game space.

My project has been devoted to the creation of a prototype of this game-like portrayal of the arena-floor competition space and the overall game state. It is also a thin prototype version of a portion of the free software that student teams will download to practice before the competition that will be useful in crafting the game to be released. It is also a way to test how the power flow algorithm will find the most efficient use of energy with the given generators, consumers, and connections of wires and transmission towers.

4 Game Display Design Requirements

There were very specific ideas that needed to be integrated into the game display. The first major part is that there needed to be a game play area in which the player would see a terrain. This terrain needed to be large but not enormous so there needed to be a way to move around the map to look around at the different parts of it. On top of this terrain the player could place consumers and generators into this area called the game world. Because of this, there needed to be some way to add different generators, consumers, and transmission structures into the game world.

Since the basis of this game is to create electrical power networks, there needed to be some way to create electrical wires between different structures. These are necessary to actually create the networks by connecting the structures into a single system. After the system is created, there needed to be some way to determine the most efficient distribution of power using the different fields of the generators and the consumers.

4.1 Different Structures

Speaking of the different structures, the generators, consumers, and transmitters needed to have certain fields for the different factors that were specific to each of them. Generators need to have a field for the maximum amount of power that it can generate. It should also have fields for the minimum power it can generate as well as the amount of power it is generating at any instantaneous moment. There should be a cost associated with building the generator as well as the cost to maintain it over time. The generator needs to keep track of how efficiently it can convert its fuel into energy. Along with this, the generator needed to keep track of the type of fuel it uses as well as the units of fuel it uses. In addition, the generator needed to keep track of

its rate of pollution. To keep each generator as an individual entity, each would need a name to specify it. Since multiple consumers could be obtaining power from a single generator, there needed to be a way to keep track of the users of a particular generator. Finally, generators need to know what node on the grid it is placed onto.

Consumers had some similar types of fields to those needed by generators but also needed to contain fields specifically for consumer information. Consumers had to be individualized so they needed names. They also need to keep track of what node they are placed onto on the grid. Fields were needed to keep track of the minimum and maximum power that the consumer could consume. It also needed to keep track of the power that is being consumed at any instantaneous moment. These are similar to the generator's maximum, minimum and instantaneous power fields. Consumers needed to keep track of the sensitivity to changes in the amount of power a consumer is obtaining. A consumer must also track the statistical variance in how much power it demands since consumers are not deterministic entities. The average power usage during the day and night are also needed for the consumer to keep a field for the mean power it needs depending on the time of day, limited here for the sake of simplicity to be either during the day or during the night.

Transmission towers were actually very simple because all that needed to be kept track of was a name to keep them as individual towers and what on node they are placed on the grid. Wires also needed to have fields kept about them to keep track of different operation pieces of information. Just as with the generators, consumers, and transmission towers the wires had id tags to allow referencing the individual instances. Each wire needed to have a maximum capacity of power that a wire would be able to transfer over distances. In addition, a wire needed to have the instantaneous amount of power it is transferring at any moment. Similar to generators, a cost

to build and a cost for maintenance needed to be available. Since wires could be constructed over a wide range of distances, the length of each wire needed to be saved somewhere. According to this length, the efficiency of the transfer of energy over a distance would change so this also needed to be kept as a field. Finally, wires needed to keep track of from what node and to what node the wire was placed.

4.2 More Information

Since a part of this game is for the user to learn about generation of energy, there needed to be a way for him or her to actually learn real life facts about energy generation and propagation.

Another screen that was necessary was an instructions screen. Since a player would need to know the controls of the game previous to beginning to play with it, this is an essential component.

There also needed to be a splash screen that connects the different screens in the game: the main game screen and the instructions screen. The splash screen would also have a way to close out of the game.

5 Components of the Empowerment Prototype

The first decision that needed to be made was in which language the prototype would be programmed. We chose to use a language that I knew, so it there would not be a learning curve in becoming skilled enough to use a new language effectively and efficiently. I have used multiple programming languages in the past. Over time, I have become skilled in using C, C++, C#, Lua, Java. Lua is a scripting language that allows quick prototyping in software development. C# is a language that is between a scripting language and an object oriented language. C is a general programming language that does not use objects. C++ is built on C and becomes more of an object oriented language by having different classes. Java is an object oriented language with highly developed libraries and borrows heavily from the syntax of C.

5.1 Game Engine Possibilities

After considering the different language choices, I also looked at a number of different game engine possibilities to use for the project.

A first possibility was the Perlenspiel engine. This is an engine programmed in Lua (at the time) that was created by Professor Brian Moriarty. Using Perlenspiel would allow for quick prototypes of what we wanted to create. Graphically, the engine is too simple to show all that we wanted on the screen because the screen is made up of only large squares called beads in the engine.

Another possibility was the GameMaker engine. This engine uses its own drag and drop system for creating games. It only uses 2 dimensional graphics called sprites. It uses its own simple scripting language to program the games.

A third possibility was the Unity engine. This engine uses 3 dimensional graphics that can be created in modeling software such as 3ds Max or Maya. Programming in Unity uses C# and Javascript for coding different events and actions.

A fourth possibility for an engine was C4. This engine uses C and C++ for coding its games. Just as with Unity, C4 uses 3 dimensional graphics that can be created in different modeling software. Its main focus is with creating first person games so this genre is the easiest to create in C4.

Another possibility was the JMonkey engine. This is a 3 dimensional game engine that uses Java for its coding. JMonkey also has many additional modules that could be downloaded and used. These include Nifty GUI which is a module that makes creating a user interface easier to do. There is also a module called Terrain Monkey which can create realistic looking terrains that could be used for creating different geography on demand.

Jgame was a sixth possibility for the project. It is a 2 dimensional engine that uses Java for programming. The web site for the engine was not very professional looking, which made it seem more amateurish than many of the other engines.

The seventh possible engine was WildTangent Web Driver. This is a 3 dimensional engine that uses Java in addition to many other languages to program games. It is normally used to create web applications but it has also been used to create stand-alone applications.

The final engine was the jMeteor engine which is a 3 dimensional engine. This engine uses Java for programming the games created in it. It is also in its alpha stage of development.

Table 1: Game Engine Qualities

Engine Choices	3D Models	Java	Standard 3D Object Formats	Active User Community	Open Source	Well Developed Documentation
Perlenspiel	No	No	None	No	Yes	Somewhat
GameMaker	No	No	None	Yes	No	Yes
Unity	Yes	No	Yes	Yes	No	Yes
C4	Yes	No	Yes	Yes	No	Somewhat
JMonkey	Yes	Yes	Yes	Yes	Yes	Yes
Jgame	No	Yes	None	Unknown	Yes	No
WildTangent Web Driver	Yes	Yes	Yes	No	Unknown	No
jMeteor	Yes	Yes	Unknown	No	Unknown	No

5.2 Deciding on an Engine

We decided that the nature of having generators, consumers, and other types of entities meant that an object-oriented programming language was needed. This would make creating classes of objects much simpler. This eliminated C and Lua as language possibilities. Java was chosen because platform portability is highly desirable for a program whose descendant may be the game software distributed to all students who engage in the Empowerment game. I have been working with Java semi-consistently during the four years of my college career. I would say that

it is the programming language that I am most comfortable with because of the extensive amount of work and projects I have written in Java. My advisor, Professor Cyganski, is also very comfortable with programming in Java which would be useful during program design and troubleshooting discussions. Choosing to use Java also eliminated many of the options for engines. Perlenspiel, GameMaker, Unity, and C4 do not use Java so they were eliminated.

Eliminating more game development engines became difficult. The first eliminated was jMeteor. Since this engine was in its alpha stages, we did not want to use it and run into a massive amount of bugs and problems trying to code with it. We decided that we wanted to use 3 dimensional object models in this application, so Jgame was eliminated as a possibility. The final two possible game engines were WildTangent Web Driver and JMonkey. I decided to evaluate the engines by reading online reviews of each to see how people who have actually developed with the engines have rated them. Using online reviews of various game engines, I looked at the scores for each engine. The scores for WildTangent WebDriver were very average (3s out of 5) in all areas of consideration. The scores for JMonkey were better than 3s (at least 4s) in most of the areas evaluated. So together with my advisor, we decided to use the JMonkey engine for this project. It supports a variety of development environments to use when creating JMonkey applications. You have the ability to use the Netbeans or Eclipse development environment. It also has its own integrated development environment (IDE) based software development kit (SDK) which can be used for coding the JMonkey applications.

5.3 JMonkey Engine

After deciding on the JMonkey Engine 3 (JME 3) as the engine that I was going to use for the project, I had to begin learning how to use it. This is when I ran into my first problem. When I first tried downloading the entire JMonkey Development Platform, it would not run

anything. I could not figure out any reason as to why this was occurring. To solve this problem, I uninstalled the entire platform. I re-downloaded the installer from the website (jmonkeyengine.org) and tried installing it again. This time the platform would correctly run the main simple application that it already has loaded as a sample.

As work on the project moved on, more problems surfaced. Many of these problems arose from the fact that I was using the alpha of the third version of the JMonkey engine. Because of this, much of the documentation was out of date as much of the documentation still only addressed version two. It seemed that a great deal changed with the alpha which made many of the tutorials and documentation useless. The ideas were the same but trying to use the same functions or methods did not prove to be equivalent to the newest version.

Halfway through the project, the beta of version three released. I tried to transfer my code into this newest version but my code could not work in the beta. The first issue was one function for setting the camera was no longer available for use. Even after setting the in-game camera in a different fashion, the code still would not compile and run. Many more pieces of behind the scenes functionality must have changed between the alpha and the beta as well. We were unable to overcome these problems with transferring to the beta version so we decided to keep working with the alpha version and work around the problems encountered.

6 Experiences

6.1 Problems

While working with the alpha of version 3 of the JMonkey Engine, many problems were encountered. One of the first problems encountered was with trying to create a ghost effect when moving objects. This required creating new versions of the textures in Photoshop to have an alpha channel. I added one but when importing and using them in the game, they did not work correctly. They looked the same as the normal textures. This was solved by showing the objects as a colored wire frame when moving them instead. Another issue was trying to get the game to pause when on any nifty-gui screen other than the HUD screen. Nifty-gui is the built in graphical user interface creator. It uses xml files to format interfaces in the form of screens. This never got implemented but it seems to work fine without this functionality.

Importing objects also became a problem. Attempting to import many different types of objects showed that next to none of the types of models actually would import. The documentation stated many of them would work. This leads to the documentation problem. Much of the documentation was still for version two of JMonkey. This proved problematic sometimes when trying to implement different aspects of the game. Searching and asking questions on the forums proved helpful in trying to figure out how to do different things when they did not work and the documentation stated they should. One example of this is setting materials. The documentation stated the generic material type was “ColorMap” for objects. Forum searches showed that the correct type to use was “Color”.

BitMapText also became a problem. It is in-game text that you can show on screen. Because of a bug in the alpha code, the font and color of the BitMapText is unable to be

changed. To fix this, I added a background of a darker color to stay behind the text so it would be easier to see.

6.2 Beta Version

Partway into C term, the beta version was released. Trying to transfer code to this proved to be a problem. The camera creation and setting was changed between the alpha and beta. The setDirection function no longer existed. I found another function called setFrame to use to set the direction of the camera in a different way, but the program would still not even open correctly in the beta. For this reason, I decided to go back to the alpha version and just try to workaround any other problems that would arise.

One problem with the camera still existed in the alpha. As objects reached the edges of the screen, they would lean drastically. This looked very unrealistic and can be seen in Figure 7: Leaning tower. This was fixed by changing the view frustrum of the camera. Instead of being set to such a wide view, I decreased the field of view to a smaller degree and moved the camera object further away from the terrain.

Figure 7: Leaning tower

6.3 Object Selection

The final major problem was encountered when trying to augment the tower implementation so it would be easier to click onto it to move it and make connections to it. To do this, I tried to create an invisible box around the tower. There is no way to add one to be the child of the other. The only way to make this work was to create a node that contained both objects. This worked and made it much easier to click on the tower. But much of my other code used the name of the tower object to move it around and save its connections and such. Sometimes the transparent box would move without the tower and vice versa. When making wire connections, it would sometimes connect to the tower and sometimes to the box. All of the problems this created were not worth fixing, so I removed the box. We then decided on making the objects into wire frames when clicked on (left or right) to know that the click was successful.

7 Prototype Design

Now, we will discuss the design of the actual prototype. This includes how the player controls the game as well as how the user interface was created. The art assets will also be discussed as well as an example of how to go through the creation process of an electrical network.

7.1 Controls

As with any game, there are controls for the player to use to interact with the game world. Since these controls are not obvious, and their use must be explained to the player so they do not have to fumble around and discover command keys and menu functions on their own.

From the splash screen shown in Figure 8: The Start-up (Splash) Screen, there are three button options that you can click on. The first button is the “Start” button. This takes you to the main game screen with the Heads-Up Display and the play area. The next button is the “Instructions” button which takes you to the screen which describes the controls used in the game as shown in Figure 9: Instructions Screen. The third button is the “Quit” button which closes the entire application.

Figure 8: The Start-up (Splash) Screen

On the instructions screen in Figure 9: Instructions Screen, you learn about controlling your view of the game world. These first controls of the game came inherently from the type of camera that was used. We decided to use the fly camera in JMonkey because of its movement abilities. The default camera is used for first person games and uses the mouse as the turning/looking mechanism. The fly camera disables this functionality and only uses keyboard buttons for movement in the three axes. The x axis, moving the camera to the left and right, is controlled by the 'A' (left) and 'D' (right) buttons. Moving along the y axis is moving up and down the map. This is controlled by using 'Q' to move up and 'Z' to move down the map. 'W' and 'S' are used to move along the Z axis. 'W' is used to zoom in and 'S' is used to zoom out.

Figure 9: Instructions Screen

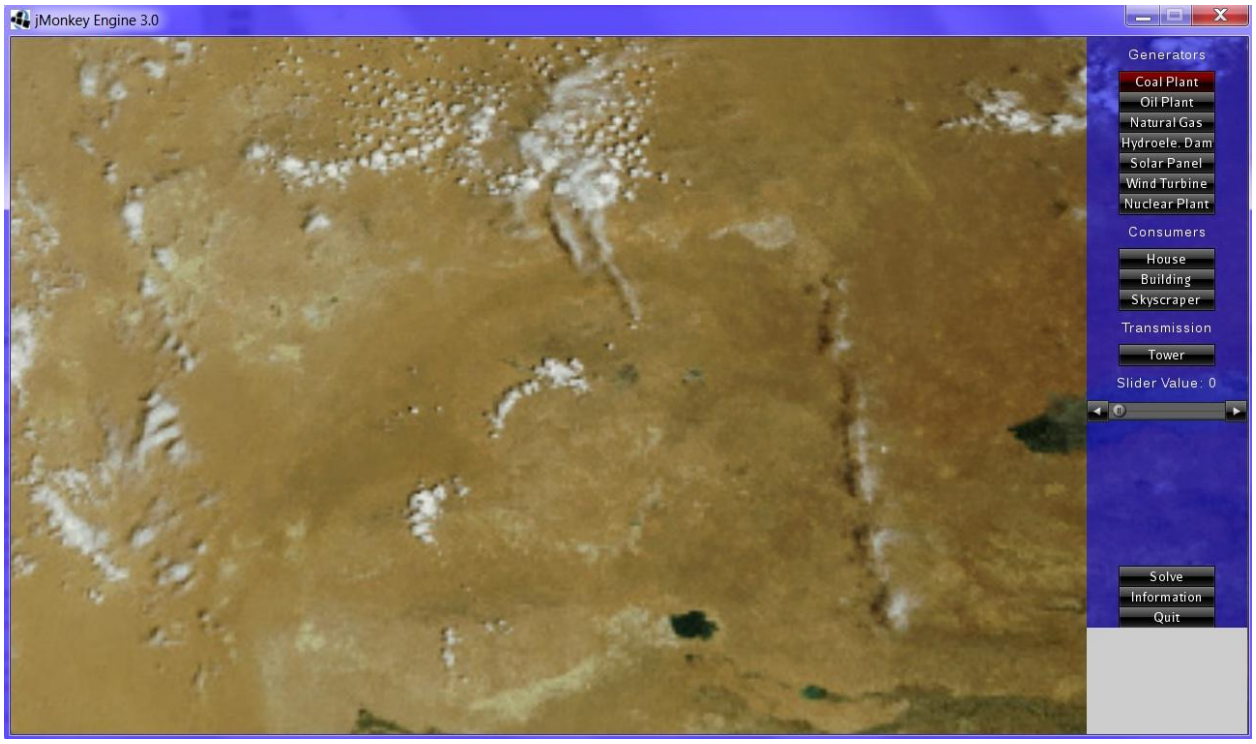

Figure 10: Game World Screen

Figure 10: is the beginning screen shown after the user clicks on the “Start” button on the start-up screen or the instruction screen. It shows the blank terrain with no objects populated in the world. This is an image of a terrain that I found by searching the Internet. It was taken from http://www.eosnap.com/tag/kyzyl-kum/ that stated on their website that any of their images can be used for educational purposes. The purpose of the terrain is to have a realistic representation of the real world that the players would be placing the objects onto instead of having a simple colored background that displays the grid on it similar to the floor arena play area. It was loaded as a texture that was placed onto a geometric plane in the game space. In the future, Terrain Monkey could be used to create a three dimensional terrain that would be even more realistic than a flat image of terrain. It could be used to create mountains, rivers and many other different types of terrain elements that would need to be worked around on the game board.

Nifty GUI is used to create the HUD. This is a module that comes with the JMonkey platform when you download and install it. It is used to create different layouts for interactive user menus and displays. An xml file is used to create all of the sections and controls to make the HUD functional. An example of the xml file for the main heads up display is shown in Appendix B. Nifty GUI is also used to create the layouts for the start up screen and the instructions screen.

It was fairly simple to learn to use and was good to work with. As can be seen in Appendix B, it provides an environment that is very similar to that of programming an html web page. It first defines the screen with a name and defines a controller for it to connect to that uses Java code to add functionality. This is because the Nifty GUI only makes the layout and the possibilities for actions very similar to the relationship between HTML and Javascript files in web sites. After creating the screen, it is subdivided into panels. These create the actual layout of the page. One layout can be seen in Figure 11: Splash screen showing panels. The panels use

height and width percentages of the pages to create different layouts. Figure 11 shows a red section which is 100% of the width while only a portion of the height so it appears at the top of the page as its own section. The blue sections lower down containing buttons only have 50% as the width so they are placed next to each other.

In the panels, controls and text can be added. Text is just text in the panel which can be aligned in multiple ways. Controls are normally buttons like the “Start and “Quit” buttons in Figure 11. These have an interact property that defines the function that will be called when the control is interacted with by the user.

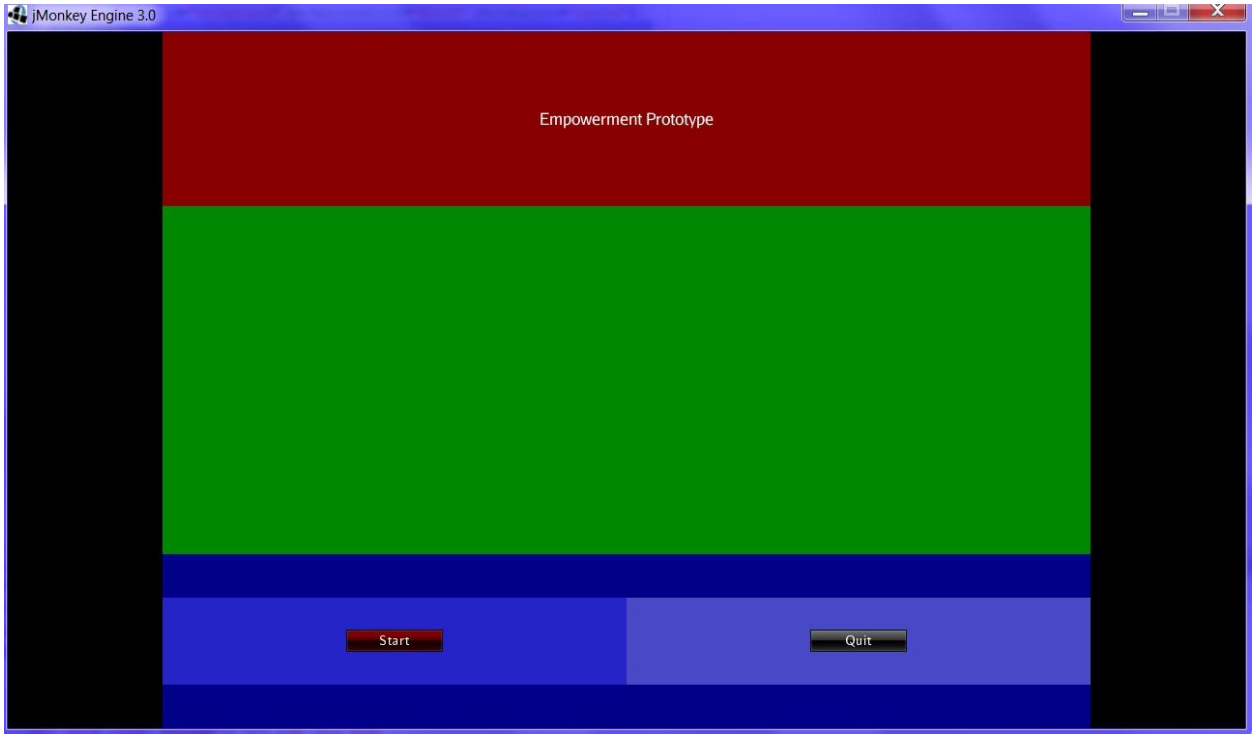

Figure 11: Splash screen showing panels

The Nifty GUI also had a large problem. An error in the alpha code would not allow pop-ups to actually pop up into the game. I got around this by creating a timer. The timer would see how long the mouse was stationary. After a determined amount of time, if the mouse was over an object, a BitMapText would appear with the information about the object.

The right-hand side of the game world screen shows the graphical user interface (GUI). It has a section of buttons for creating generators. The generators that can be created are a coal plant, an oil plant, a natural gas plant, a hydroelectric dam, a solar panel, a wind turbine, and a nuclear power plant. This could be expanded in the future or changed to a single menu item that could bring up a menu of generators to choose from. The next section contains buttons used to create the consumers. For the purposes of the prototype this is allowed, but in the actual version of the game used in the competition, the consumers would be stationary and placed beforehand by the program according to the scenario. The consumers that can be created currently are a house, a building, and a skyscraper. The next section of buttons contains the transmission objects. The only transmission structure currently available to be created is a transmission tower. In the future, the transmission section would also contain a button to create substations and various types of towers. The next part of the GUI has a slider bar and shows the current value of the setting on the slider bar. This is used when creating an object to determine its maximum power value or when resetting the power value of an object. Both of these operations are discussed below. There is a fair amount of blank space leftover on the sidebar HUD element. This can be utilized later with more buttons for different types of energy generators or consumers and even some other transmission objects. The next button is the “Solve” button. This solves the current electrical power network that has been created. This will be discussed in more detail below. The final button is the “Quit” button which cleanly exits the JMonkey application.

As just stated above, creating generators and consumers is accomplished using the buttons on the heads up display on the right side of the screen as seen in Figure 10: . You can control the max power for generators and the required power for consumers using the slider in the HUD. Just choose the value you would like the next generator or consumer to have and then click on the button to create the object with that value. Moving objects is pretty intuitive. You left click and hold over an object that you wish to move. If the user has successfully clicked on an object, it will have become a blue wire frame as shown in Figure 12. This shows that the user is actually clicking on an object so it is completely obvious.

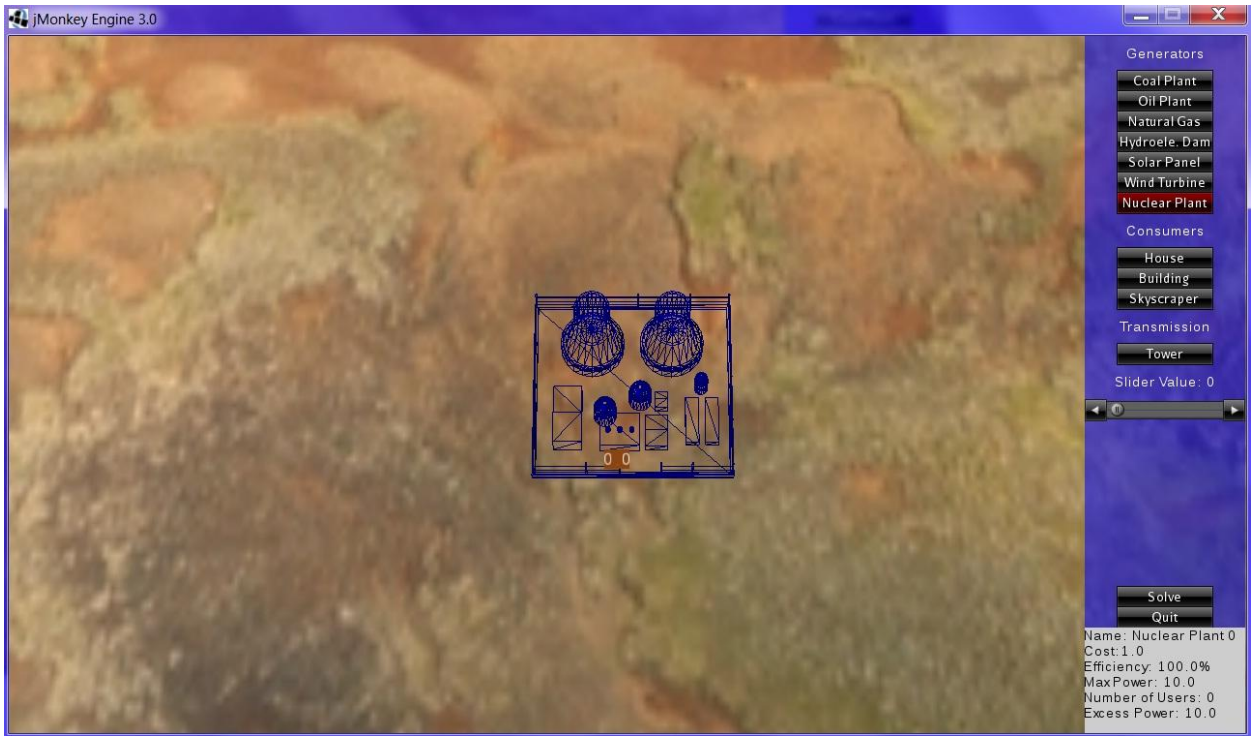

Figure 12: On left clicking an object, selection is indicated by its 3D representation becoming a blue wireframe model. The object will now move with the mouse.

You can then move the mouse anywhere on the screen, and stop holding the mouse button. The object will then move to the spot where the mouse cursor is located as seen in Figure

13. The need to assertively indicate that an object had been selected came about specifically with the tower object. As seen in Figure 31: Transmission Tower Object Model, the tower object has a lot of empty space compared to other objects and the actual parts of the model are very small on screen. This created a problem arising from the difficulty in being able to click on an element of it. Clicking on it required much more precision that any of the other objects. I tried to create an invisible box around the tower to make it easier to click on, but this broke more functionality than it fixed as described earlier. Thus, it was removed.

Figure 13: Once the selected object has been moved, it is de-selected by releasing the left mouse button upon which its solid representation reappears.

Creating power lines works in a similar fashion but with some differences. You right click on an object but do not hold it. If you successfully clicked on the object, it will become an orange wire frame as shown in Figure 14. This feature is implemented for the same reason as

indicated above: for the ease of left-clicking for object movement.

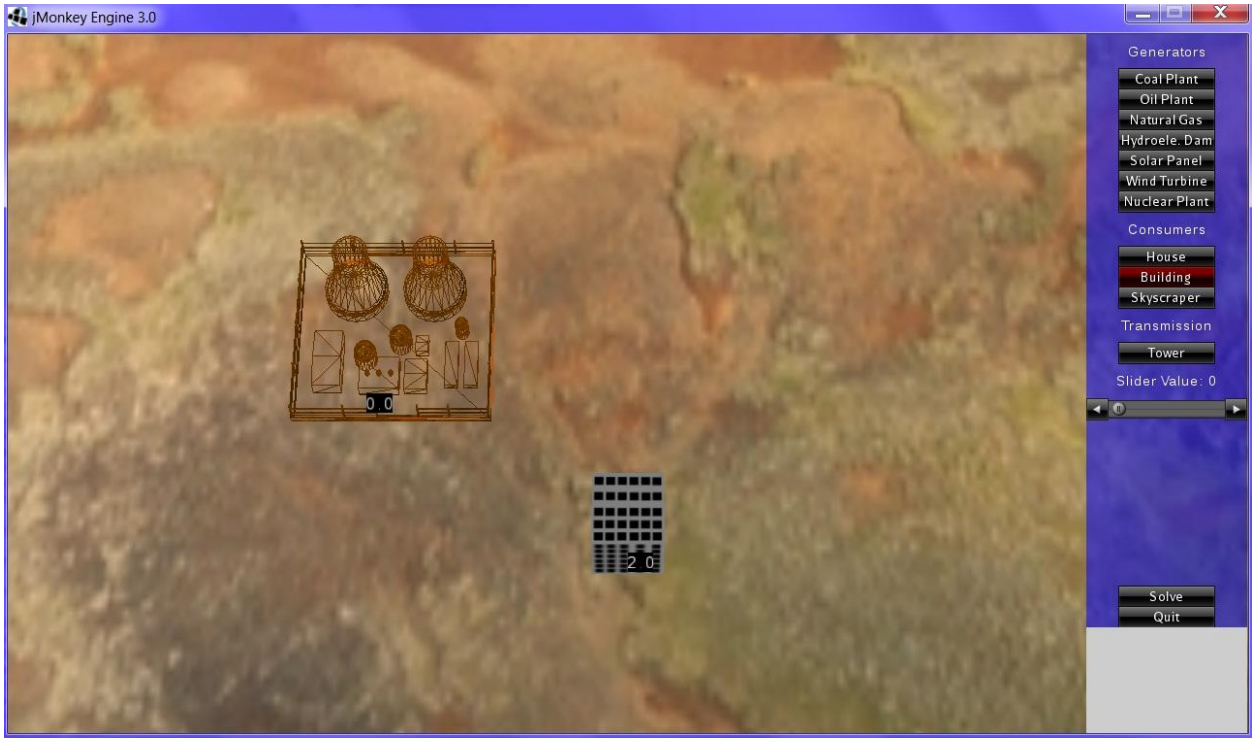

Figure 14: On right clicking an object, selection is indicated by its 3D representation becoming an orange wireframe model. A transmission line will be created between this and the next selected object.

You then right click on the next object that you would like the first object to be connected to. This will create a blue line between them which is the graphical representation of the wire between them as seen in Figure 15.

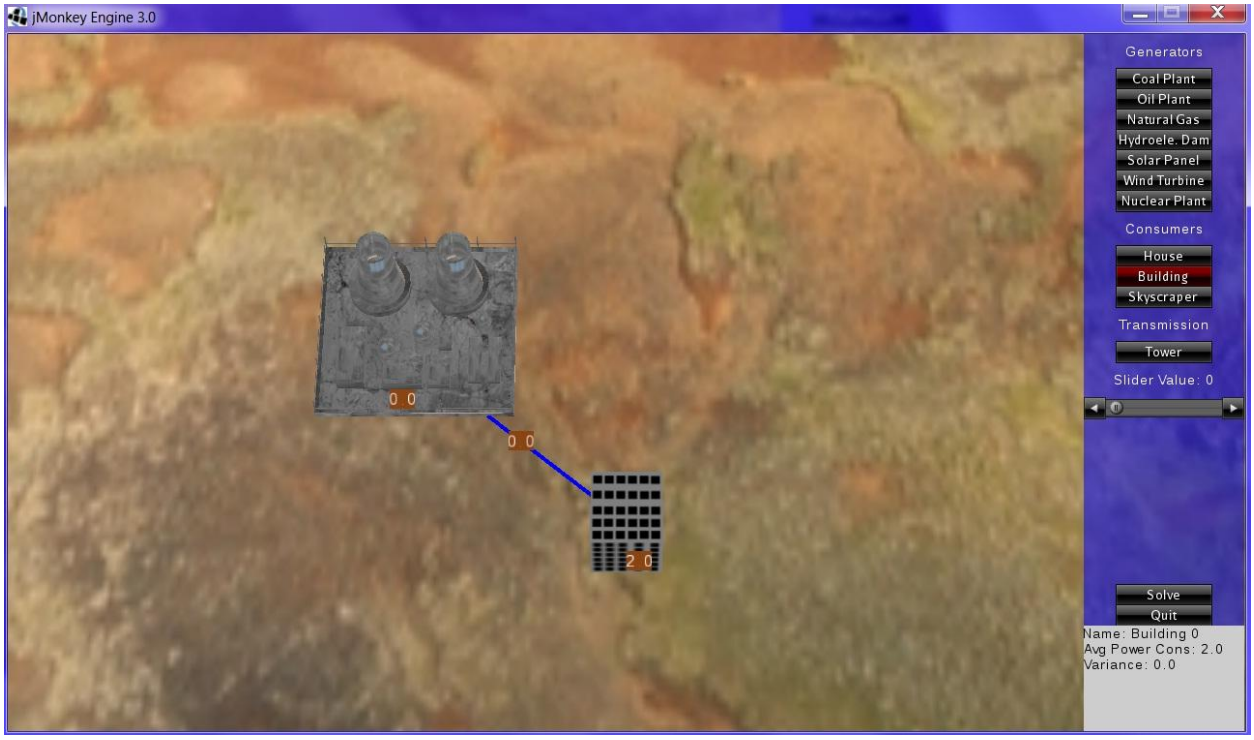

Figure 15: Once a second object has been selected, the solid representation reappears and a wire is created connecting the two objects.

You can also reset the max power for generators or the required power for consumers. To reset this value you place the mouse cursor over the object you wish to reset. You choose the value on the slider that you want the new max or required power to be. Then you press the 'R' key. This updates the object value to the new value from the slider. Figure 16 shows the max power of the nuclear plant as being 10 MW, which is the default hard coded in its creation. Figure 17 shows the max power updated with the current slider value of 43. These two figures also show the pop up and inset features of the game. Both appear when the mouse if held over an object for approximately three seconds without moving. They both also show the same information. The difference being that the pop up disappears when the mouse moves again while

the inset remains until the next time it is updated when the mouse is held still over an object for three seconds.

Figure 16: Before updating the object, the max power is seen as 10 MW.

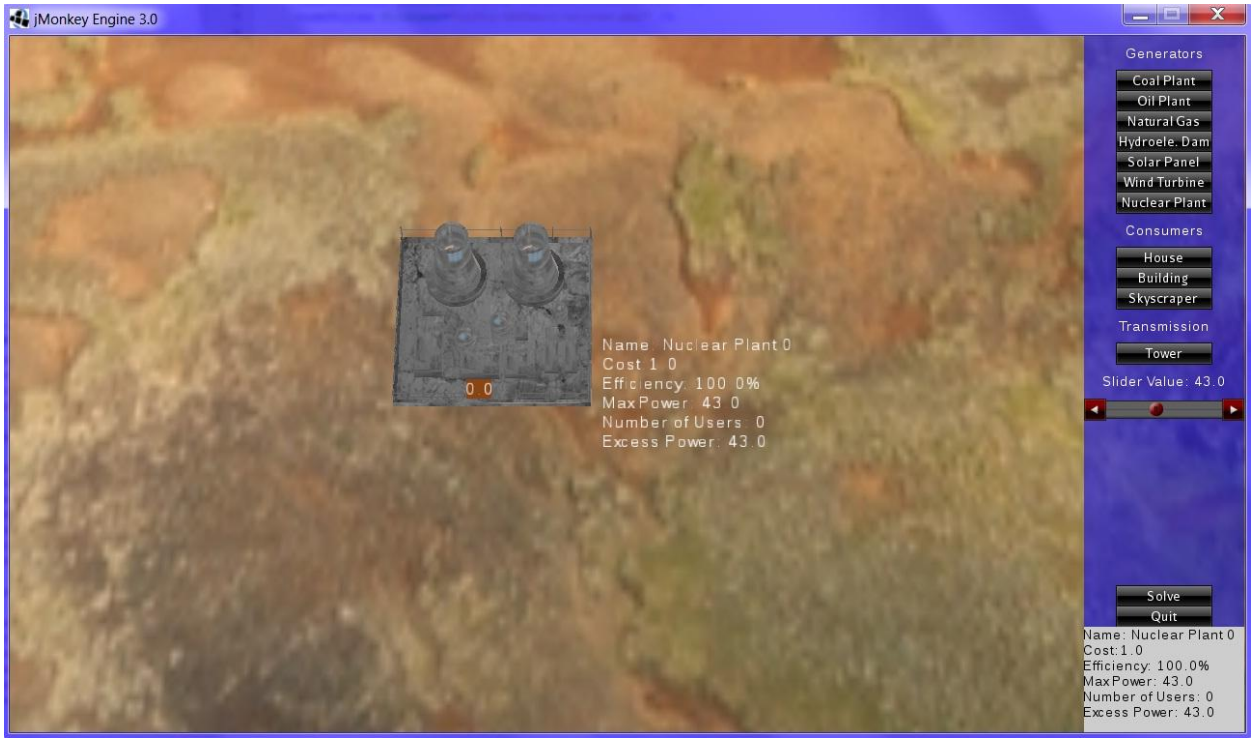

Figure 17: After setting the slider to a value and updating the object, the object's max power is 43 MW.

When you have a full electrical power system created in the game, you can press the solve button to determine the most efficient distribution of power to be created by each generator and to be sent over each line that still gives enough power to each consumer to satisfy their need. In Figure 20 there are signs on all of the electrical wires showing how much power is travelling through them. Generators show how much power they are producing while consumers display how much power they require. There are also two situations that could occur if there is a problem. The first problem that could occur is that a wire may reach its capacity. In this case, the wire will turn red and not transfer any more power. This is shown in Figure 18. The other problem that could occur is that a consumer does not get enough power to meet its demand. In this case, the sign for that consumer will show the percentage of brown out that is occurring.

This percentage is how much power is still requires over the total amount it requires. This is shown in Figure 19.

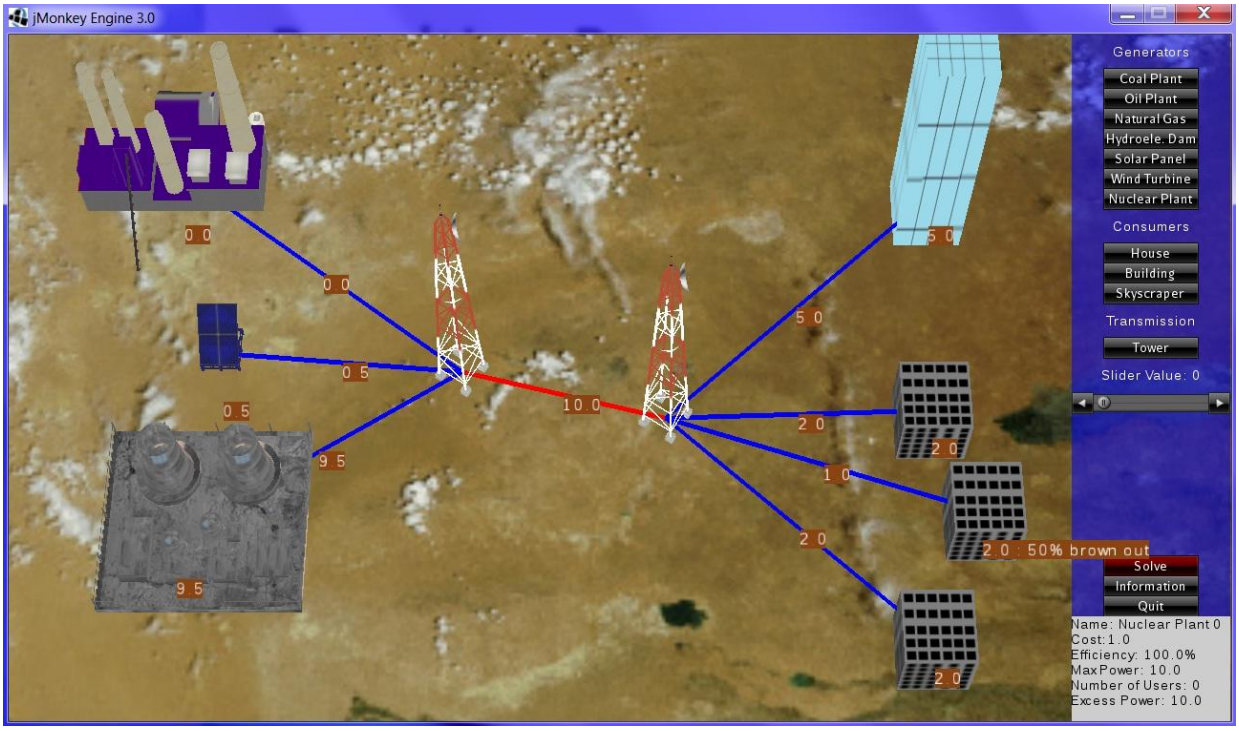

Figure 18: The connection between the towers is incapable of supporting the level of power demanded by consumers. This is indicated by its color turning red.

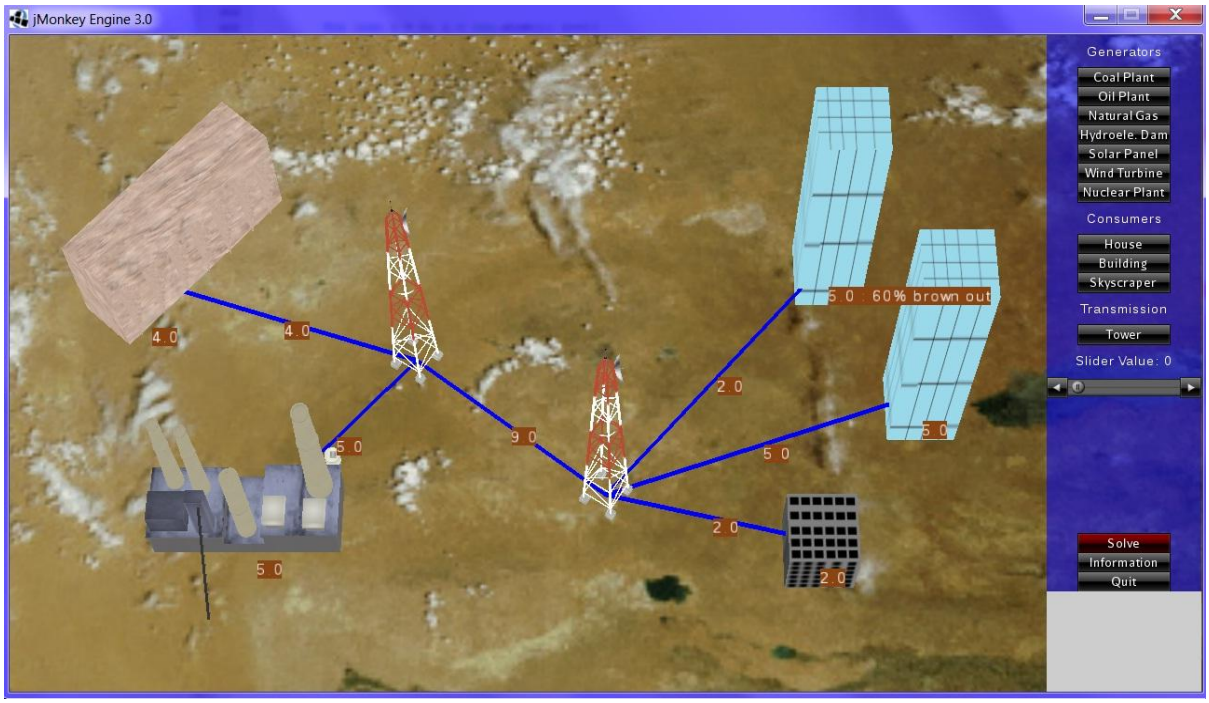

Figure 19: A consumer is not receiving its required power. This is indicated by the brownout percentage.

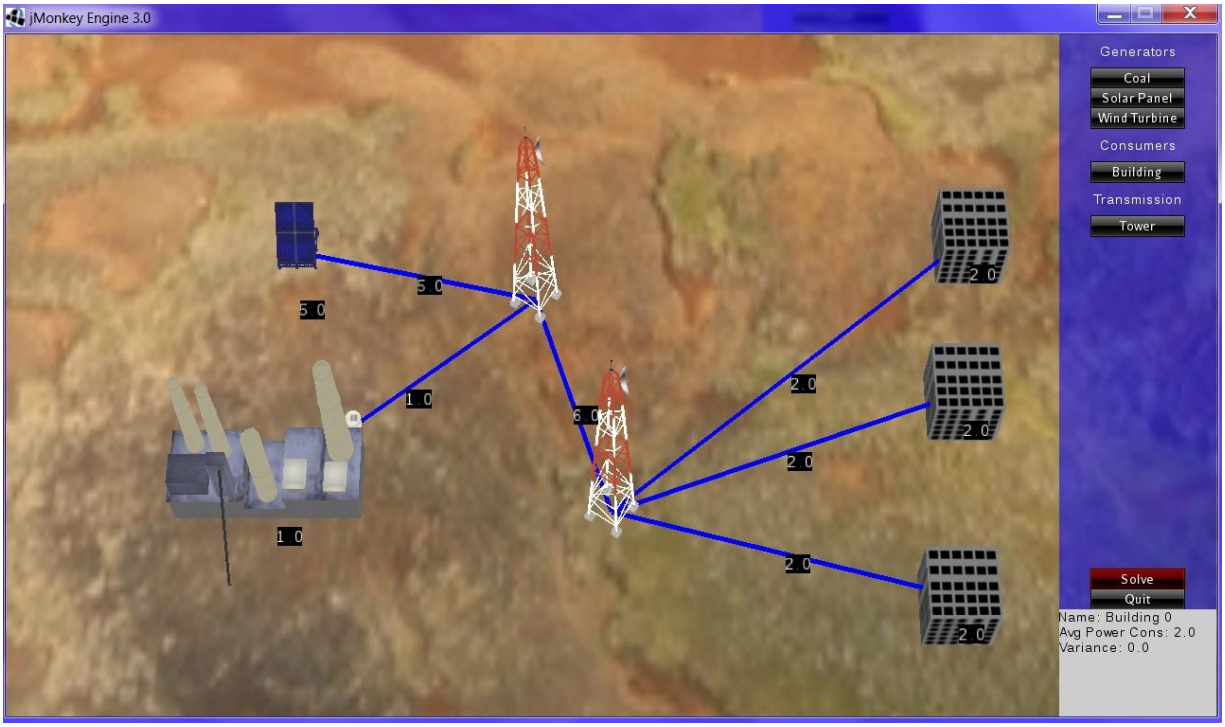

Figure 20: Example of a solved network with no problems

The final action that can be taken is to delete objects. To delete an object, you place the mouse cursor over the object you want to eliminate. Then you press the 'E' key. This deletes the object and also removes any lines that were attached to it. There is also a "Quit" button in the HUD to close the application.

7.2 3D Models and Textures

After some time working with JMonkey engine, we learned that some of the documentation on their website (jmonkeyengine.org and jmonkeyengine.com) is not entirely accurate. Some even contradicts itself. Looking for the types of models that could be imported into the engine the documentation indicated that all of the general types of object models could be imported without any problem. Trying to import most model types proved to be a problem. Autodesk Maya and 3D Studio Max files would not import into the engine at all. The only model files that would import into the system were Wavefront object files (.obj).

A set of models were then created to support the basic types of power system entities we wished to support as described above. These are listed below along with who created them. I enlisted two of my friends to aid in the creation of the 3D models to be used in the game prototype.

The Coal Plant model was created by Ethan Lawrence. The image of the coal plant in game can be seen in Figure 21: Coal Plant Object Model.

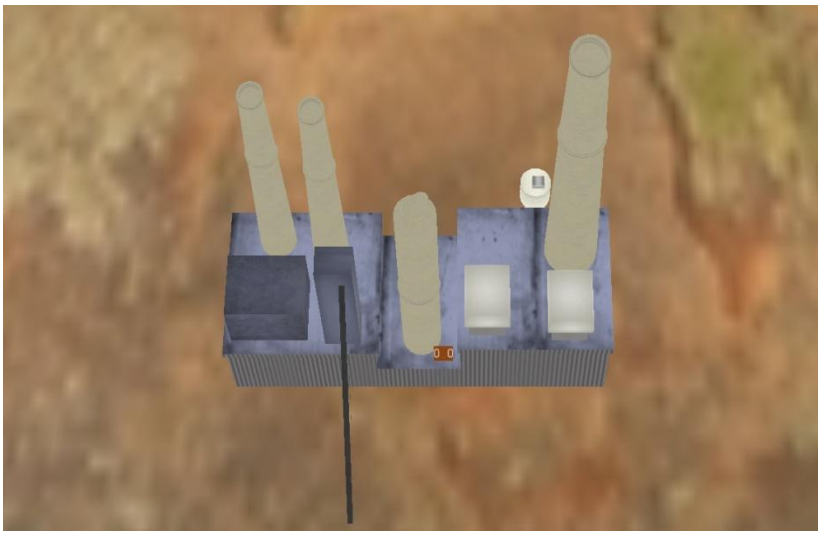

Figure 21: Coal Plant Object Model

The Oil Plant model is the same as coal plant, but I modified texture to be a different color. The image of the oil plant in game can be seen in Figure 22: Oil Plant Object Model.

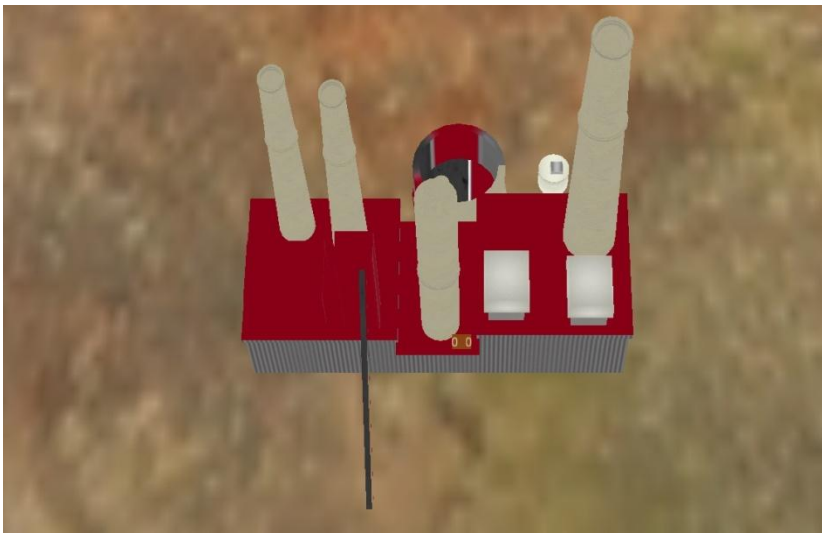

Figure 22: Oil Plant Object Model

The Natural Gas Plant model is the same as coal plant, but I modified texture to be a different color. The image of the natural gas plant in game can be seen in Figure 23: Natural Gas Plant Object Model.

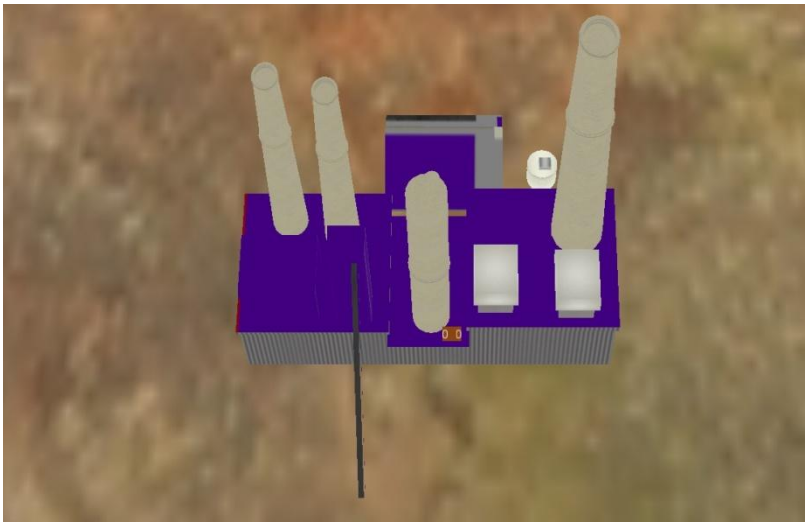

Figure 23: Natural Gas Plant Object Model

I created the Hydroelectric Dam in Maya. The image of the hydroelectric dam in game can be seen in Figure 24: Hydroelectric Dam Object Model.

Figure 24: Hydroelectric Dam Object Model

The Solar Panel was created by Ashley McGuane. The image of the solar panel in game can be seen in Figure 25: Solar Panel Object Model.

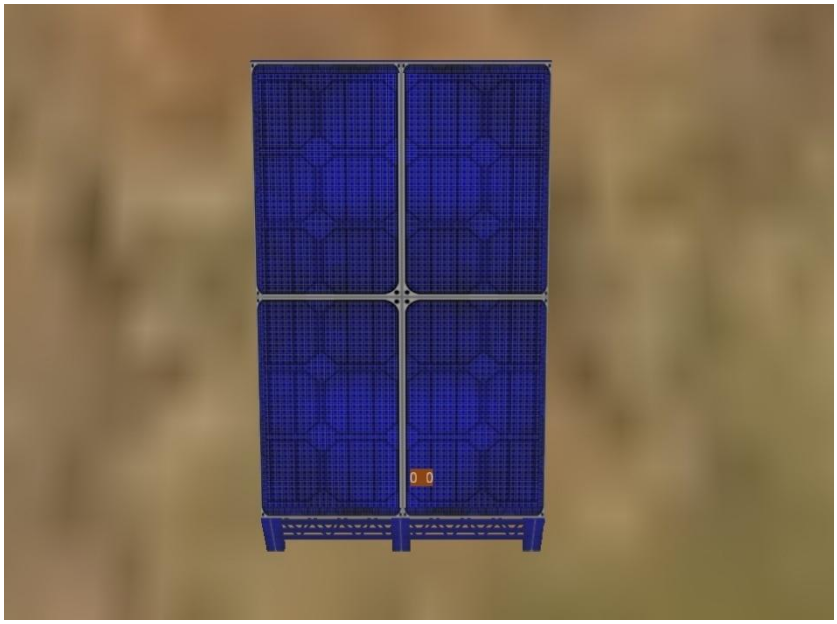

Figure 25: Solar Panel Object Model

The Wind Turbine was created by Ashley McGuane. The image of the wind turbine in game can be seen in Figure 26: Wind Turbine Object Model.

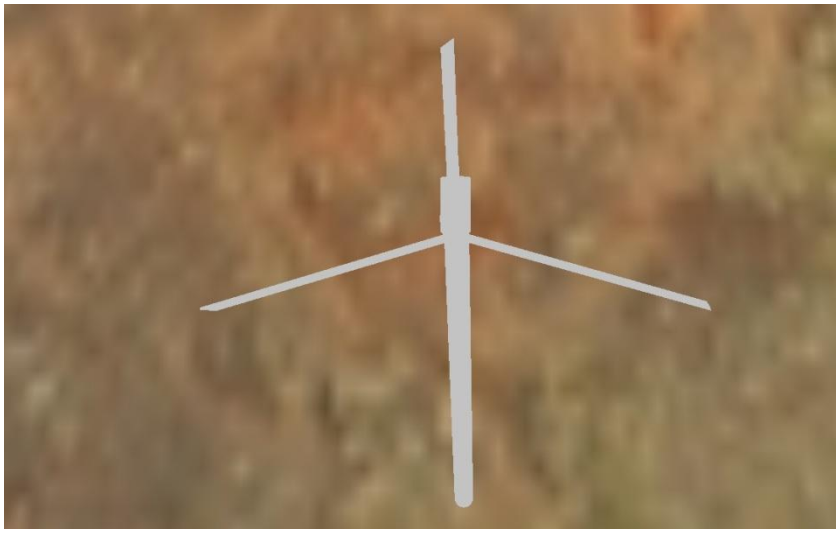

Figure 26: Wind Turbine Object Model

The Nuclear Plant was created by Ashley McGuane. The image of the nuclear power plant in game can be seen in Figure 27: Nuclear Power Plant Object Model.

Figure 27: Nuclear Power Plant Object Model

The House was made by me in the JMonkey engine as just a box geometry. The image of the house in game can be seen in Figure 28: House Object Model.

Figure 28: House Object Model

The Building was made by me in the JMonkey engine as just a box geometry. The image of the building in game can be seen in Figure 29: Building Object Model.

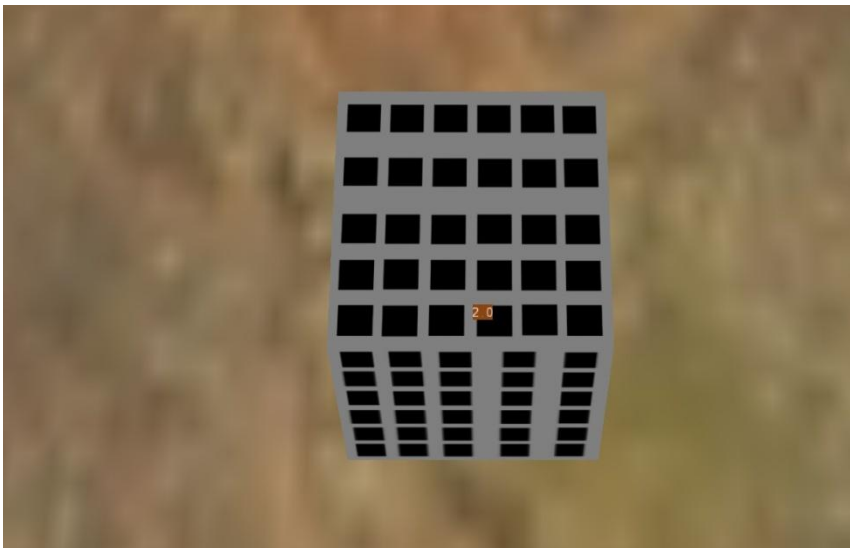

Figure 29: Building Object Model

The Skyscraper was made by me in the JMonkey engine as just a box geometry. The image of the skyscraper in game can be seen in Figure 30: Skyscraper Object Model.

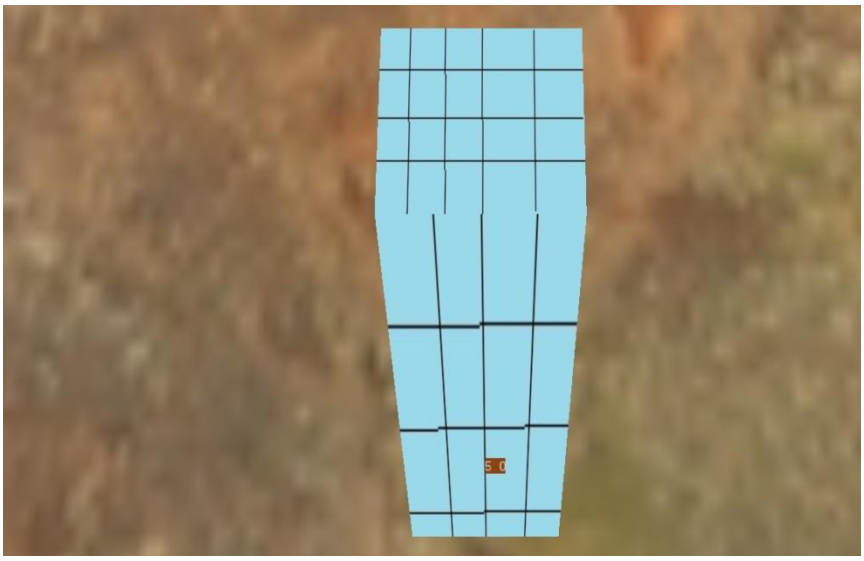

Figure 30: Skyscraper Object Model

The Transmission Tower was created by Ethan Lawrence. The image of the transmission tower in game can be seen in Figure 31: Transmission Tower Object Model.

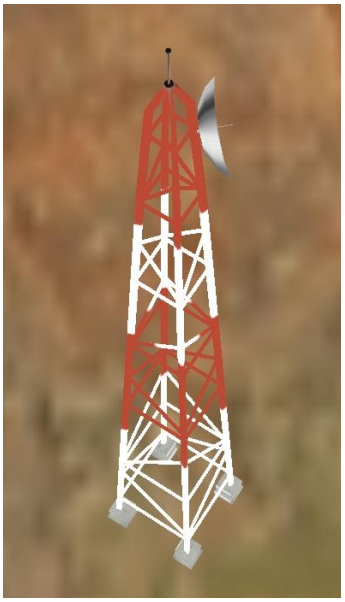

Figure 31: Transmission Tower Object Model

7.3 Example Application

In the following screens, we can see the creation and solving of a network. In Figure 32, all of the objects that will be involved in the network have been added to the game screen using the buttons in the HUD on the right side of the screen. Next, all of the connections are added in the network. As in Figure 33, the connections have been added to create a full network. The hydroelectric dam and the coal plant have been connected to the first tower. This tower has been connected to the second one. The second tower is connected to the two buildings and one skyscraper. After the network is created, the solve button is clicked. The aftermath of this can be seen in Figure 34. In this figure, 4 MW of power is being generated by the dam and 5 MW is being generated by the coal plant. This is being transferred though the transmission lines to the first tower. The 9 MW of power is transferred from the first to the second tower. The second tower sends 2 MW of power to each of the buildings and 5 MW of power to the skyscraper.

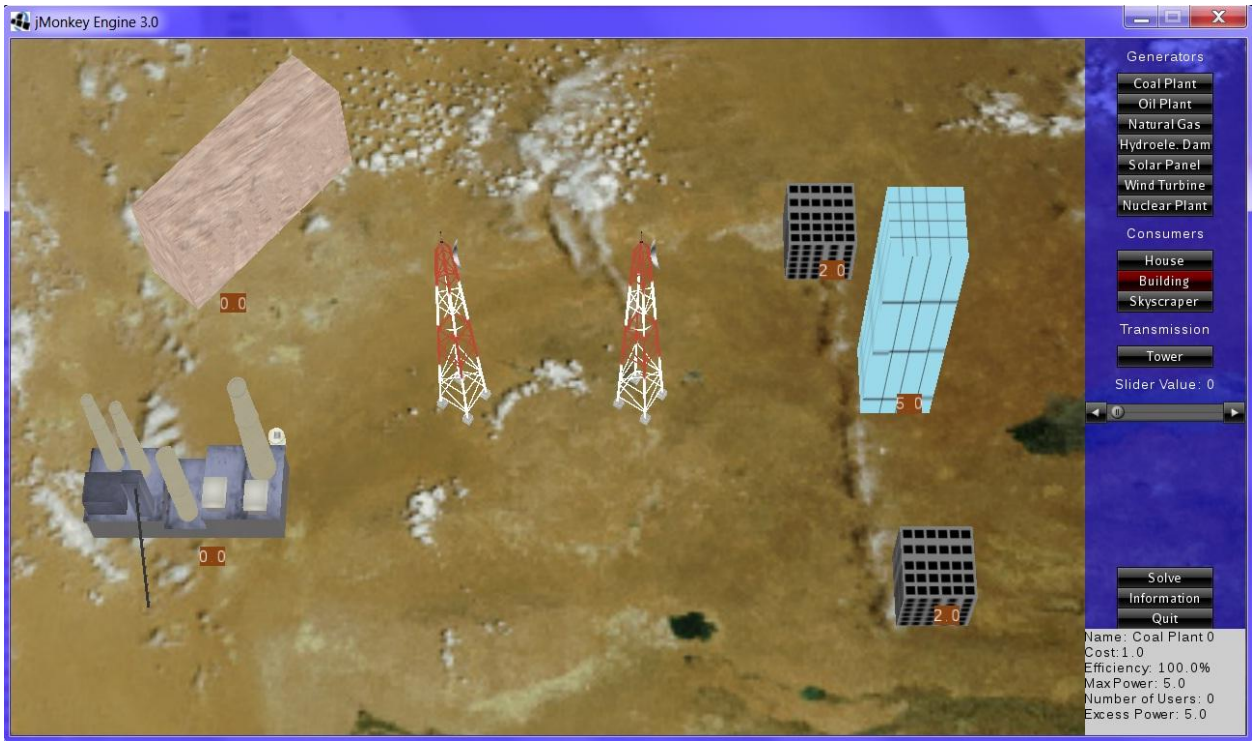

Figure 32: Objects added

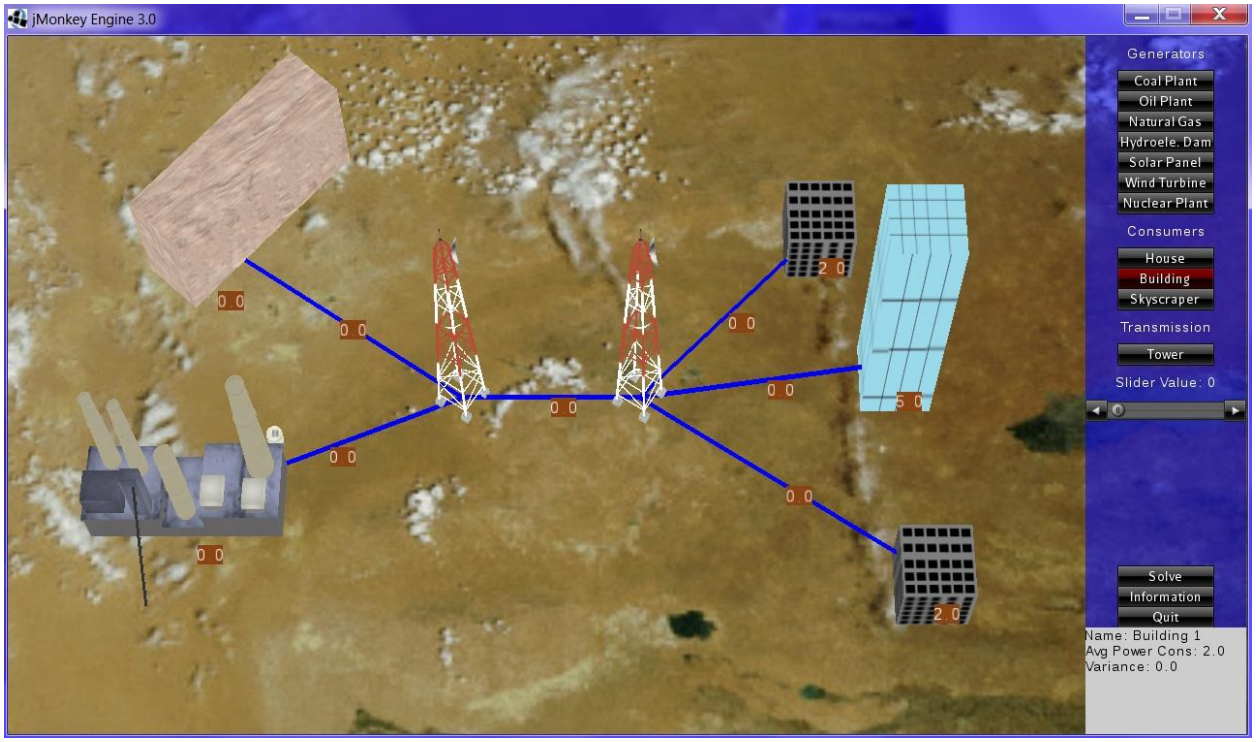

Figure 33: Connections added

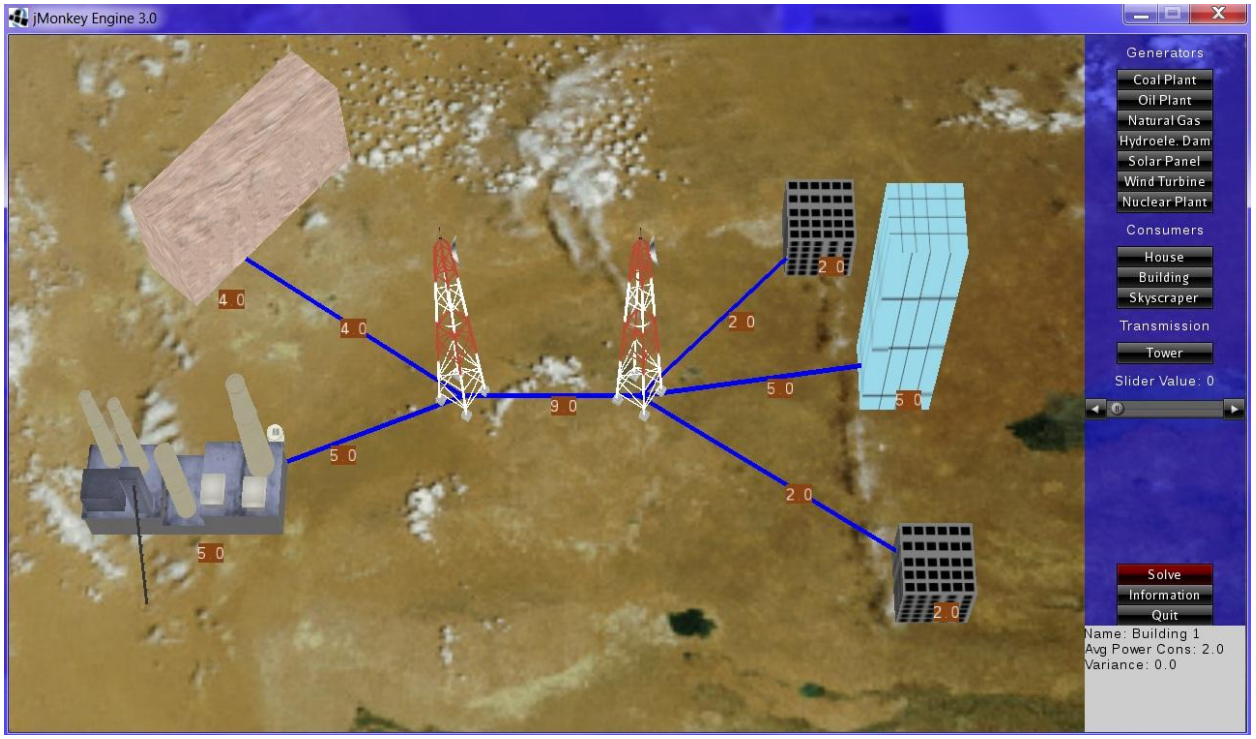

Figure 34: Network solved

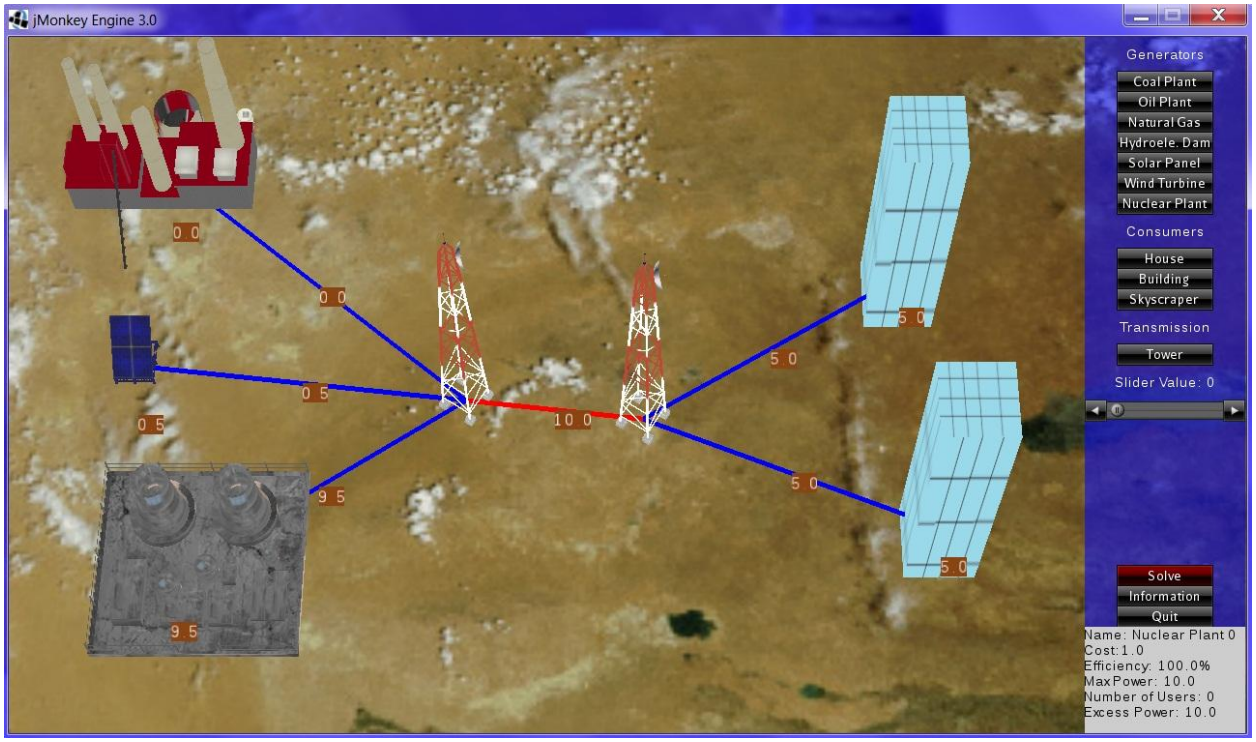

Figure 35: Wire Capacity Reached

In Figure 35, the solar panel is the most cost efficient, so it generates all the power it can which id 0.5 MW. The nuclear power plant generates 9.5 MW of power. The oil plant does not produce any power. This power is transferred to the first tower. Then, 10 MW is transferred to the second tower. Since the capacity of the wire is 10 MW, it turns red because it has reached capacity. The power is then split evenly between the two skyscrapers. In this example, the maximum capacity is 10 MW so that is all that is generated by the most cost efficient generators.

In Figure 36, all of the generators are at full capacity. The wind turbine, natural gas plant, and solar panel are generating 1, 1, and 0.5 MW of power respectively. The 2.5 MW of power is transferred to the first tower and then to the second. From the second tower, 2 MW are transmitted to the first building. The second building only receives 0.5 MW of power. It requires 2 MW. Since it is not getting all the power it needs, it has a brownout. This percentage is

calculated by dividing the amount of power it is not receiving over how much it is requiring. In this case that is 1.5 / 2 which equals 75%. So it is only receiving 25% of the power it requires so it has a 75% brownout. The final building has 100% brownout because it is not even connected to the network.

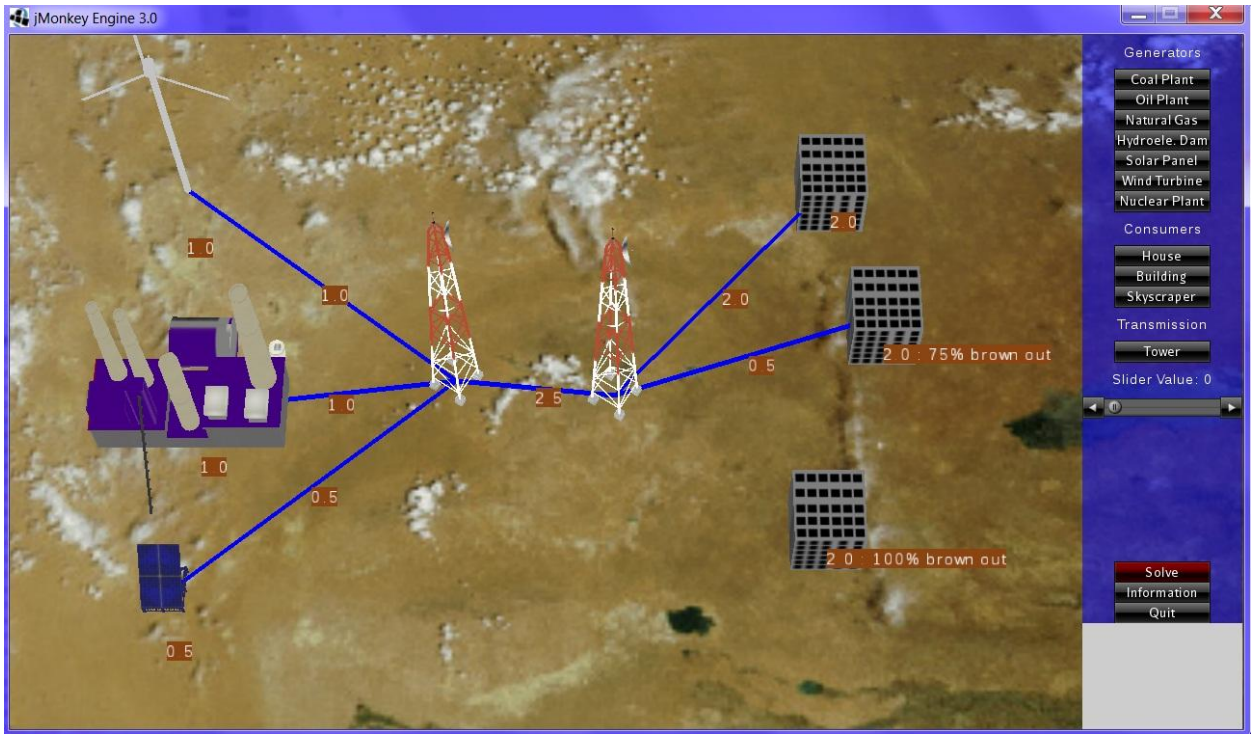

Figure 36: Multiple Brownouts in Network

8 Conclusion

The final version of the Empowerment prototype program has all of the base functionality working. The player has the ability to create the 3 dimensional objects corresponding to generators, consumers, and transmission towers. Connections can be made between these different objects to create a fully connected electrical power network. The player also has the ability to change the main property of each generator and consumer. Finally, there is the ability to solve the network for the most cost optimum way to transfer the required energy from the generators, through the network, and to the consumers.

The main problems encountered in this project revolved around using the alpha version of the JME software. There was a lack of clear documentation for this new version. The majority of the documentation described version two. While some of this still applied to the alpha of version three, much of it has been changed. The other main problems had to do with the broken functionality that was discovered with the alpha throughout the project. These were avoided through different methods described,

In the future, much more functionality would be added. First, we would want to have 3 dimensional terrain rather than a flat 2 dimensional image of terrain. This would increase the realism of the game. Another addition would be zoning. This means that different objects would only be able to be placed in certain areas. For example, a hydroelectric dam would only be able to be placed on a source of running water so there would be something to turn the turbines to generate the energy.

More fields of objects would be changeable. This would enable a wider variety of differences between copies of the same object type. More objects would also be available in the

future. For example, substations would be created under the transmission category. There would also be a need for more variety in consumers and possibly more different generator types. The cost function for solving the network for the optimum cost transfer of energy would become more complex. There would be additions of environmental factors to change the priority of different types of energy. Consumer sensitivity would also be taken into account. For example, a hospital is much more sensitive in losing its energy because it contains people who need to be kept alive with the power. The final large change would be an AI game climate. This would automate user variations. During the course of the day, the power requirements of users would change mimicking the variations between daytime and nighttime energy requirements, heavier loads in summer, etc. It would also entail resource variation like a lack of sun for solar panels, a lack of wind for wind farms, and a drought for hydroelectric dams.

This project has come very far from the start, but also has a lot of ideas to be added still. With fewer problems arising from the incomplete and unstable nature of the current JME implementation, more could have completed by the end of the project.

9 Appendices

9.1 Appendix A: Framework Analysis

Perlenspiel uses Lua for scripting which allows for quick prototypes but the graphical capabilities would not be sufficient for what we are doing and what we need.

GameMaker uses a drag and drop system for creating games. It is a 2D engine. Since it uses its own simple “language”, I do not think it would be well suited for the needs of the project because of the extreme simplicity. I do not believe it is complex enough for an MQP.

Unity is a 3D game engine that uses C# and Javascript for coding. I think that this engine could work nicely for what we need for the project. I have seen projects that do similar things as what we are going to be creating.

C4 is a 3D game engine that uses C++ or C for coding. In my experience, this engine is rather difficult to work with, so I do not think it would work well for what we need. It does not make 2D style games like we are creating very well. It is mainly an engine for creating first person games.

jMonkey Engine (jME) is a 3D engine that uses Java for coding. It seems to have features we will be able to use along with extra features that go beyond what we need. I am only unsure of how well it will actually do what we need since I have not used it previously.

Jgame is a 2D engine that uses Java for coding. It seems to be good for what we need but I am unsure of how well it really works since the website does not look very professional. But I will not write it off just because of that.

Others:

WildTangent WebDriver is an engine that uses Java, in addition to many other languages, for coding. The ratings on the engine are average (only 3 stars out of 5) so I am unsure about its ability to do what we need quickly and easily.

jMeteor is a 3D game engine that uses Java for coding. It is only in the Alpha stage of development so I would not recommend using it.

9.2 Appendix B: XML Example

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- Created by Timothy Kolek

 MQP 2011 - 2012-->

<nifty xmlns="http://nifty-gui.sourceforge.net/nifty-1.3.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

      xsi:schemaLocation="http://nifty-gui.sourceforge.net/nifty-1.3.xsd http://nifty-
gui.sourceforge.net/nifty-1.3.xsd">

   <useControls filename="nifty-default-controls.xml" />

   <useStyles filename="nifty-default-styles.xml" />


   <screen id="hud" controller="Empowerment.HUDScreenController">

<!--       <layer id="background" backgroundColor="#000f">

       </layer>-->

       <layer id="foreground" backgroundColor="#0000" childLayout="horizontal">

          <panel id="panel_left" width="87%" height="100%" childLayout="vertical">

<!--              <panel id="panel_left1" width="25%" height="25%" childLayout="vertical"
backgroundColor="#3456">

                 <control id="inset" name="label" textHAlign="left" textVAlign="top" color="#000"
text="" width="100%" height="100%" ></control>

             </panel>-->

        spacer

     </panel>

     <panel id="panel_right" width="13%" height="100%" childLayout="vertical"

     backgroundColor="#00f8">

        <panel id="panel_rightA" width="100%" height ="5%" childLayout="center">

           <text text="Generators" font="Interface/Fonts/Liberation.fnt" width="100%"
height="100%" />

        </panel>
```

```
        <panel id="panel_right1" width="100%" height ="3%" childLayout="center">
            <control name="button" label="Coal Plant" id="CoalButton" align="center"
valign="center" visibleToMouse="true">
                <interact onClick="makeCoal()"/>
            </control>
        </panel>
        <panel id="panel_right11" width="100%" height ="3%" childLayout="center">
            <control name="button" label="Oil Plant" id="OilButton" align="center"
valign="center" visibleToMouse="true">
                <interact onClick="makeOil()"/>
            </control>
        </panel>
        <panel id="panel_right12" width="100%" height ="3%" childLayout="center">
            <control name="button" label="Natural Gas" id="GasButton" align="center"
valign="center" visibleToMouse="true">
                <interact onClick="makeGas()"/>
            </control>
        </panel>
        <panel id="panel_right13" width="100%" height ="3%" childLayout="center">
            <control name="button" label="Hydroele. Dam" id="DamButton" align="center"
valign="center" visibleToMouse="true">
                <interact onClick="makeDam()"/>
            </control>
        </panel>
        <panel id="panel_right6" width="100%" height ="3%" childLayout="center">
            <control name="button" label="Solar Panel" id="SolarButton" align="center"
valign="center" visibleToMouse="true">
                <interact onClick="makeSolar()"/>
            </control>
```

```
    </panel>
    <panel id="panel_right7" width="100%" height ="3%" childLayout="center">
      <control name="button" label="Wind Turbine" id="WindButton" align="center"
valign="center" visibleToMouse="true">
        <interact onClick="makeWind()"/>
      </control>
    </panel>
    <panel id="panel_right8" width="100%" height ="3%" childLayout="center">
      <control name="button" label="Nuclear Plant" id="NuclearButton" align="center"
valign="center" visibleToMouse="true">
        <interact onClick="makeNuclear()"/>
      </control>
    </panel>
    <panel id="panel_rightB" width="100%" height ="5%" childLayout="center">
      <text text="Consumers" font="Interface/Fonts/Liberation.fnt" width="100%"
height="100%" />
    </panel>
    <panel id="panel_right10" width="100%" height ="3%" childLayout="center">
      <control name="button" label="House" id="HouseButton" align="center"
valign="center" visibleToMouse="true">
        <interact onClick="makeHouse()"/>
      </control>
    </panel>
    <panel id="panel_right2" width="100%" height ="3%" childLayout="center">
      <control name="button" label="Building" id="BuildingButton" align="center"
valign="center" visibleToMouse="true">
        <interact onClick="makeBuilding()"/>
      </control>
    </panel>
```

```
        <panel id="panel_right9" width="100%" height ="3%" childLayout="center">
            <control name="button" label="Skyscraper" id="SkyscraperButton" align="center"
valign="center" visibleToMouse="true">
                <interact onClick="makeSkyscraper()"/>
            </control>
        </panel>
        <panel id="panel_rightC" width="100%" height ="5%" childLayout="center">
            <text text="Transmission" font="Interface/Fonts/Liberation.fnt" width="100%"
height="100%" />
        </panel>
        <panel id="panel_right3" width="100%" height ="3%" childLayout="center">
            <control name="button" label="Tower" id="TowerButton" align="center"
valign="center" visibleToMouse="true">
                <interact onClick="makeTower()"/>
            </control>
        </panel>

        <panel id="panel_right spacer2" width="100%" height ="5%" childLayout="center">
            <control id="sliderValue" name="label" textHAlign="center" textVAlign="center"
font="Interface/Fonts/Liberation.fnt" text="Slider Value: 0" width="100%" height="100%"
></control>
        </panel>

<!--    temp panel to put inset at bottom-->
        <panel id="panel_right spacer" width="100%" height ="27%" childLayout="center">
            <control id="slider" name="horizontalSlider" width="100%"
buttonStepSize="1.0"></control>
        </panel>
```

```
        <panel id="panel_right4" width="100%" height ="3%" childLayout="center">
            <control name="button" label="Solve" id="SolveButton" align="center"
valign="center" visibleToMouse="true">
                <interact onClick="goSolve()"/>
            </control>
        </panel>
        <panel id="panel_right5" width="100%" height ="3%" childLayout="center">
            <control name="button" label="Quit" id="QuitButton" align="center" valign="center"
visibleToMouse="true">
                <interact onClick="quitGame()"/>
            </control>
        </panel>
        <panel id="panel_right inset" width="100%" height ="16%" childLayout="center"
backgroundColor="#CDCDCD">
            <effect>
              <onActive name="textSize" startSize="5" endSize="5" > </onActive>
            </effect>
            <control id="inset" name="label" textHAlign="left" textVAlign="top"
font="Interface/Fonts/Liberation.fnt" color="#000" text="" width="100%" height="100%"
></control>
        </panel>
      </panel>
       </layer>


   </screen>
<!--    <popup id="popupCons" childlayout="center" backgroundColor="#000a">
```

```
        <panel id="popupPanel" width="100%" height="100%" childLayout="center">

        <text text="Average Power Consumed" font="Interface/Fonts/Liberation.fnt"
width="100%" height="100%" />

        <control id="sliderH" name="horizontalSlider" width="100px" />

        </panel>

    </popup>-->

</nifty>
```